

Proceedings of the
2010 International Symposium on
Symbolic and Algebraic Computation

ISSAC 2010

25-28 July 2010, Munich Germany

Stephen M. Watt, Editor

**The Association for Computing Machinery
2 Penn Plaza, Suite 701
New York, New York 10121-0701**

ACM COPYRIGHT NOTICE. Copyright © 2010 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept., ACM, Inc., fax +1 (212) 869-0481, or permissions@acm.org.

For other copying of articles that carry a code at the bottom of the first or last page, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, +1-978-750-8400, +1-978-750-4470 (fax).

Notice to Past Authors of ACM-Published Articles

ACM intends to create a complete electronic archive of all articles and/or other material previously published by ACM. If you have written a work that was previously published by ACM in any journal or conference proceedings prior to 1978, or any SIG Newsletter at any time, and you do NOT want this work to appear in the ACM Digital Library, please inform permissions@acm.org, stating the title of the work, the author(s), and where and when published.

ACM ISBN: 978-1-4503-0150-3

Foreword

The International Symposium on Symbolic and Algebraic Computation is the premier conference spanning all areas of research in symbolic mathematical computing. The series has a long history, established in 1966 and operating under the ISSAC name since 1988. This year's meeting is the 35th occurrence and is held at the Technische Universität München.

With a subject that has been so thoroughly studied for half a century some might ask whether the main questions have been answered and whether any important challenges remain. Nothing could be farther from the present exciting state of affairs! A quick glance through these proceedings will reveal a subject that is more vibrant than ever. In some ways, we are today experiencing a golden age in symbolic computing: On one hand, we are studying a wider range of mathematical problems and we have deeper algorithmic insight into the central questions than ever before. On the other hand, the scale and nature of computing hardware that is widely available make asymptotically fast and parallel algorithms of immediate practical interest. Not only are these computational problems very interesting in their own right, their solution has a significant practical impact, affecting the millions of users of free and commercial computer algebra packages. Surely there has never been a more interesting time in symbolic mathematical computation.

ISSAC 2010 brings together a good number of the world's most active researchers in the area for a period of four days. As has become our tradition, the meeting features invited presentations, tutorials, contributed research papers, software presentations and a poster session for works in progress. In this way, the participants are able to keep up with a broad range of areas and to present work at different stages of maturity.

The invited presentations touch both on central topics in computer algebra and highly relevant nearby areas:

Evelyne Hubert: *Algebraic Invariants and their Differential Algebras*

Siegfried M. Rump: *Verification Methods: Rigorous Results using Floating-Point Arithmetic*

Ashish Tiwari: *Theory of Reals for Verification and Synthesis of Hybrid Dynamical Systems*

We are grateful that these distinguished speakers have agreed to speak at our meeting.

The ISSAC tutorials have always been popular. They are intended to make new areas accessible to students and practitioners in other areas of the field. This year we are fortunate to have tutorials by three truly talented expositors:

Moulay A. Barkatou: *Symbolic Methods for Solving Systems of Linear Ordinary Differential Equations*

Jürgen Gerhard: *Asymptotically Fast Algorithms for Modern Computer Algebra*

Sergey P. Tsarev: *Transformation and Factorization of Partial Differential Systems with Applications to Stochastic Systems*

At ISSAC 2010, in a departure from the usual practice, the tutorials carry no registration fee. We are eager to see what effect this has.

As usual, the main body of the conference consists of contributed research papers. A call for papers was circulated one year prior to the meeting, inviting contributions in all areas of computer algebra and symbolic mathematical computation. These included:

Algorithmic aspects: exact and symbolic linear, polynomial and differential algebra; symbolic-numeric, homotopy, perturbation and series methods; computational geometry, group theory and number theory; summation, recurrence equations, integration, solution of ODE & PDE; symbolic methods in other areas of pure and applied mathematics; theoretical and practical aspects, including general algorithms, techniques for important special cases, complexity analyses of algebraic algorithms and algebraic complexity;

Software aspects: design of packages and systems; data representation; software analysis; considerations for modern hardware, e.g., current memory and storage technologies, high performance systems and mobile devices; user interface issues, including collaborative computing and new methods for input and manipulation; interfaces and use with systems for, e.g., document processing, digital libraries, courseware, simulation and optimization, automated theorem proving, computer aided design and automatic differentiation;

Application aspects: applications that stretch the current limits of computer algebra algorithms or systems, use computer algebra in new areas or new ways or apply it in situations with broad impact.

In response, 110 submissions were received and considered. These were reviewed by members of the Program Committee and a wide range of external reviewers. In all, 349 reviews were obtained and every paper received between 3 and 5 reviews. PC members could neither participate in nor see the discussions relating to papers with which they had conflicts of interest. Following the PC deliberations, 45 contributed research papers were accepted for presentation at the conference and inclusion in these proceedings.

These proceedings present the contributed research papers in the order of presentation. They are grouped loosely by topic in a manner that fits the conference schedule. Other presentation groupings might provide somewhat more scientific coherence, but could not be accommodated for practical reasons.

Running a meeting such as ISSAC consumes the efforts of many people. We would like to express our gratitude to all those who have contributed. We first thank the invited speakers and tutorial presenters for agreeing to participate. We thank the authors of the research papers for contributing their work. We are extremely grateful to the members of the PC and the army of external reviewers for their careful work on a tight schedule. We thank Andrei Voronkov for his assistance with EasyChair, Peter Horn for designing the cover and Vadim Mazalov for his substantial assistance in preparing these proceedings. We especially thank the entire local organizing team who have worked hard to make this conference enjoyable and productive. Finally, on behalf of our entire community, we thank the Deutsche Forschungsgemeinschaft and Maplesoft for their generous financial support.

Wolfram Koepf
General Chair

Stephen M. Watt
Program Committee Chair

Ernst W. Mayr
Local Arrangements Chair

June 24, 2010

Table of Contents

Invited Presentations

Algebraic Invariants and Their Differential Algebras	1
<i>Evelyne Hubert</i>	
Verification Methods: Rigorous Results using Floating-Point Arithmetic	3
<i>Siegfried M. Rump</i>	
Theory of Reals for Verification and Synthesis of Hybrid Dynamical Systems	5
<i>Ashish Tiwari</i>	

Tutorials

Symbolic Methods for Solving Systems of Linear Ordinary Differential Equations ...	7
<i>Moulay A. Barkatou</i>	
Asymptotically Fast Algorithms for Modern Computer Algebra.....	9
<i>Jürgen Gerhard</i>	
Transformation and Factorization of Partial Differential Systems: Applications to Stochastic Systems	11
<i>S. P. Tsarev</i>	

Contributed Papers

<i>Gröbner Bases</i>	
A New Incremental Algorithm for Computing Groebner Bases	13
<i>Shuhong Gao, Yinhua Guan and Frank Volny</i>	
Degree Bounds for Gröbner Bases of Low-Dimensional Polynomial Ideals	21
<i>Ernst W. Mayr and Stephan Ritscher</i>	
A New Algorithm for Computing Comprehensive Gröbner Systems	29
<i>Deepak Kapur, Yao Sun and Dingkang Wang</i>	
<i>Differential Equations</i>	
Finding all Bessel Type Solutions for Linear Differential Equations with Rational Function Coefficients.....	37
<i>Mark van Hoeij and Quan Yuan</i>	
Simultaneously Row- and Column-Reduced Higher-Order Linear Differential Systems	45
<i>Moulay Barkatou, Carole El Bacha and Eckhard Pflügel</i>	
Consistency of Finite Difference Approximations for Linear PDE Systems and Its Algorithmic Verification.....	53
<i>Vladimir Gerdt and Daniel Robertz</i>	

CAD and Quantifiers

Computation with Semialgebraic Sets Represented by Cylindrical Algebraic Formulas	61
<i>Adam Strzeboński</i>	

Black-Box/White-Box simplification and Applications to Quantifier Elimination	69
<i>Christopher Brown and Adam Strzeboński</i>	

Parametric Quantified SAT Solving	77
<i>Thomas Sturm and Christoph Zengler</i>	

Differential Algebra I

A Method for Semi-Rectifying Algebraic and Differential Systems using Scaling Type Lie Point Symmetries with Linear Algebra	85
<i>François Lemaire and Ash Ürgüplü</i>	

Absolute Factoring of Non-holonomic Ideals in the Plane	93
<i>Dima Grigoriev and Fritz Schwarz</i>	

Algorithms for Bernstein-Sato Polynomials and Multiplier Ideals	99
<i>Christine Berkesch and Anton Leykin</i>	

Polynomial Algebra

Global Optimization of Polynomials Using Generalized Critical Values and Sums of Squares	107
<i>Feng Guo, Mohab Safey El Din and Lihong Zhi</i>	

A Slice Algorithm for Corners and Hilbert-Poincaré Series of Monomial Ideals	115
<i>Bjarke Hammersholt Røne</i>	

Composition Collisions and Projective Polynomials	123
<i>Joachim von zur Gathen, Mark Giesbrecht and Konstantin Ziegler</i>	

Decomposition of Multivariate Polynomials	131
<i>Jean-Charles Faugère, Joachim von zur Gathen and Ludovic Perret</i>	

Seminumerical Techniques

NumGfun: a Package for Numerical and Analytic Computation with D-finite Functions	139
<i>Marc Mezzarobba</i>	

Chebyshev Interpolation Polynomial-Based Tools for Rigorous Computing	147
<i>Nicolas Brisebarre and Mioara Joldes</i>	

Blind Image Deconvolution via Fast Approximate GCD	155
<i>Zijia Li, Zhengfeng Yang and Lihong Zhi</i>	

Polynomial Integration on Regions Defined by a Triangle and a Conic	163
<i>David Sevilla and Daniel Wachsmuth</i>	

Geometry

- Computing the Singularities of Rational Space Curves** 171
Xiaoran Shi and Falai Chen
- Solving Schubert Problems with Littlewood-Richardson Homotopies** 179
Frank Sottile, Ravi Vakil and Jan Verschelde
- Triangular Decomposition of Semi-Algebraic Systems** 187
Changbo Chen, James H. Davenport, John P. May, Marc Moreno Maza, Bican Xia and Rong Xiao

Differential Algebra II

- When Can We Detect that a P-Finite Sequence is Positive?** 195
Manuel Kauers and Veronika Pillwein
- Complexity of Creative Telescoping for Bivariate Rational Functions** 203
Alin Bostan, Shaoshi Chen, Frédéric Chyzak and Ziming Li
- Partial Denominator Bounds for Partial Linear Difference Equations** 211
Manuel Kauers and Carsten Schneider

Polynomial Roots and Solving

- Real and Complex Polynomial Root-finding with Eigen-Solving and Preprocessing** 219
Victor Y. Pan and Ai-Long Zheng
- Computing the Radius of Positive Semidefiniteness of a Multivariate Real Polynomial via a Dual of Seidenberg's Method** 227
Sharon Hutton, Erich Kaltofen and Lihong Zhi
- Random Polynomials and Expected Complexity of Bisection Methods for Real Solving** 235
Ioannis Z. Emiris, André Galligo and Elias Tsigaridas
- The DMM bound: Multivariate (Aggregate) Separation Bounds** 243
Ioannis Z. Emiris, Bernard Mourrain and Elias Tsigaridas

Theory and Applications

- Solving Bezout-Like Polynomial Equations for the Design of Interpolatory Subdivision Schemes** 251
Costanza Conti, Luca Gemignani and Lucia Romani
- Computing Loci of Rank Defects of Linear Matrices using Gröbner Bases and Applications to Cryptology** 257
Jean-Charles Faugère, Mohab Safey El Din and Pierre-Jean Spaenlehauer
- Output-Sensitive Decoding for Redundant Residue Systems** 265
Majid Khonji, Clément Pernet, Jean-Louis Roch, Thomas Roche and Thomas Stalinski

Linear Algebra

A Strassen-Like Matrix Multiplication Suited for Squaring and Higher Power Computation	273

Marco Bodrato

Computing Specified Generators of Structured Matrix Inverses	281

Claude-Pierre Jeannerod and Christophe Moulleron

Yet Another Block Lanczos Algorithm: How to Simplify the Computation and Reduce Reliance on Preconditioners in the Small Field Case	289

Wayne Eberly

Linear Recurrences and Difference Equations

Liouvillian Solutions of Irreducible Second Order Linear Difference Equations	297

Mark van Hoeij and Giles Levy

Solving Recurrence Relations using Local Invariants	303

Yongjae Cha, Mark van Hoeij and Giles Levy

On Some Decidable and Undecidable Problems Related to q-Difference Equations with Parameters	311

Sergei Abramov

Arithmetic

Iterative Toom-Cook Methods for Very Unbalanced Long Integer Multiplication ..	319

Alberto Zanoni

An In-Place Truncated Fourier Transform and Applications to Polynomial Multiplication	325

David Harvey and Daniel S. Roche

Randomized NP-Completeness for for p-adic Rational Roots of Sparse Polynomials in One Variable	331

Martín Avendaño, Ashraf Ibrahim, J. Maurice Rojas and Korben Rusek

Software Systems

Easy Composition of Symbolic Computation Software: A New Lingua Franca for Symbolic Computation	339

Kevin Hammond, Peter Horn, Alexander Konovalov, Steve Linton, Dan Roozemon, Abdallah Al Zain and Phil Trinder

Symbolic Integration At Compile Time in Finite Element Methods	347

Karl Rupp

Fast Multiplication of Large Permutations for Disk, Flash Memory and RAM	355

Vlad Slavici, Xin Dong, Daniel Kunkle and Gene Cooperman

Author Index	363

ISSAC 2010 Organization

ISSAC Steering Committee

André Galligo (Chair)	Université de Nice, France
Elkedagmar Heinrich	Fachgruppe Computeralgebra
Elizabeth Mansfield	SIGSAM
Michael Monagan	Simon Fraser University, Canada
Yosuke Sato	JSSAC
Franz Winkler	RISC Linz, Austria

Conference Organizing Committee

General Chair	Wolfram Koepf	Kassel, Germany
Program Committee Chair	Stephen M. Watt	London, Ontario, Canada
Local Arrangements Chair	Ernst W. Mayr	München, Germany
Poster Committee Chair	Ilias Kotsireas	Waterloo, Ontario, Canada
Software Exhibits Chair	Michael Monagan	Vancouver, Canada
Tutorials Chair	Sergei Abramov	Moscow, Russia
Publicity Chair	Peter Horn	Kassel, Germany
Treasurer	Thomas Hahn	München, Germany

Program Committee

Moulay Barkatou	Université de Limoges, France
Alin Bostan	INRIA, France
Chris Brown	US Naval Academy, USA
James Davenport	University of Bath, UK
Jean-Guillaume Dumas	Université Joseph Fourier, France
Wayne Eberly	University of Calgary, Canada
Bettina Eick	TU Braunschweig, Germany
Jean-Charles Faugère	UPMC and INRIA, France
Michael Kohlhase	Jacobs University, Germany
Laura Kovács	Vienna University of Technology, Austria
Ziming Li	Chinese Academy of Sciences, China
Elizabeth Mansfield	University of Kent, UK
B. David Saunders	University of Delaware, USA
Éric Schost	University of Western Ontario, Canada
Ekaterina Shemyakova	RISC-Linz, Austria
Thomas Sturm	Universidad de Cantabria, Spain
Carlo Traverso	Università di Pisa, Italy
Stephen Watt (Chair)	University of Western Ontario, Canada
Kazuhiro Yokoyama	Rikkyo University, Japan
Lihong Zhi	Chinese Academy of Sciences, China

Poster Committee

Ilias Kotsireas (Chair)
Markus Rosenkranz
Yosuke Sato
Eva Zerz

Wilfrid Laurier University, Canada
University of Kent, Great Britain
Tokyo University of Science, Japan
RWTH Aachen, Germany

External Reviewers

John Abbott
Sergei Abramov
Victor Adamchik
Martin Albrecht
Dhavide Aruliah
David H. Bailey
Jean-Claude Bajard
Peter Baumgartner
Alexandre Benoit
Dario A. Bini
Paola Boito
Sylvie Boldo
Delphine Boucher
Russell Bradford
Michael Brickenstein
Massimo Caboara
Bob Caviness
Bruce Char
Howard Cheng
Jin-San Cheng
Frédéric Chyzak
Thomas Cluzeau
Svetlana Cojocaru
Gene Cooperman
Robert Corless
Carlos D'Andrea
Xavier Dahan
Mike Dewar
Daouda Diatta
Philippe Elbaz-Vincent
M'hammed El Kahoui
Ioannis Z. Emiris
William Farmer
Claudia Fassino
Sándor Fekete
Ruyong Feng
Laurent Fousse
Josep Freixas
Anne Frühbis-Krüger

Mitsushi Fujimoto
André Galligo
Xiao-Shan Gao
Mickaël Gastineau
Joachim von zur Gathen
Thierry Gautier
Keith Geddes
Patrizia Gianni
Mark Giesbrecht
Pascal Giorgi
Laureano Gonzalez-Vega
Iavn Graham
Kevin Hammond
William Hart
David Harvey
Mark van Hoeij
Jerome Hoffman
Derek Holt
Max Horn
Qing-Hu Hou
Evelyne Hubert
Alexander Hulpke
Hiroyuki Ichihara
Claude-Pierre Jeannerod
Tudor Jebelean
David Jeffrey
Jeremy Johnson
Françoise Jung
Erich Kaltofen
Chandra Kambhamettu
Deepak Kapur
Manuel Kauers
Achim Kehrein
Denis Khmelnov
Alexander Kholosha
Kinji Kimura
Simon King
Jürgen Klüners
Wolfram Koepf

Alexander Konovalov
Ilias Kotsireas
Christoph Koutschan
Werner Krandick
Heinz Kredel
Martin Kreuzer
Alexander Kruppa
Benoit Lacelle
Christoph Lange
Aless Lasaruk
Daniel Lazard
Wen-Shin Lee
Bas Lemmens
Viktor Levandovskyy
Anton Leykin
Guiqing Li
Daniel Lichtblau
Steve Linton
Austin Lobo
Florian Lonsing
Salvador Lucas
Frank Lübeck
Montserrat Manubens
Mircea Marin
John P. May
Scott McCallum
Guy McCusker
Guillaume Melquiond
Marc Mezzarobba
Johannes Middeke
Maurice Mignotte
Yasuhiko Minamide
Niels Möller
Michael Monagan
Antonio Montes
Teo Mora
Marc Moreno Maza
Guillaume Moroz
Bernard Mourrain

Mircea Mustata
 Katsusuke Nabeshima
 Kosaku Nagasaka
 George Nakos
 Winfried Neun
 Masayuki Noro
 Eamonn O'Brien
 Takeshi Ogita
 François Ollivier
 Takeshi Osoekawa
 Alexey Ovchinnikov
 Victor Y. Pan
 Maura Paterson
 Clément Pernet
 Ludovic Perret
 John Perry
 Marko Petkovšek
 Veronika Pillwein
 Mihai Prunescu
 González Pérez
 Florian Rabe
 Silviu Radu
 Stefan Ratschan
 Greg Reid
 Guénaél Renault
 Nathalie Revol
 Lorenzo Robbiano
 Jean-Louis Roch
 Enric Rodríguez Carbonell
 J. Maurice Rojas

Markus Rosenkranz
 Fabrice Rouillier
 Olivier Ruatta
 Rosario Rubio
 Mohab Safey El Din
 Massimiliano Sala
 Bruno Salvy
 Yosuke Sato
 Peter Scheiblechner
 Carsten Schneider
 Hans Schönemann
 Wolfgang Schreiner
 Johann Schuster
 Fritz Schwarz
 Markus Schweighofer
 Robin Scott
 Werner M. Seiler
 Hiroshi Sekigawa
 Vikram Sharma
 Takafumi Shibuta
 Naoyuki Shinohara
 Kiyoshi Shirayanagi
 Igor Shparlinski
 Michael F. Singer
 Mate Soos
 Volker Sorge
 Eduardo Sáenz-de-Cabezón
 Allan Steel
 Doru Stefanescu
 Damien Stehlé

Arne Storjohann
 Adam Strzeboński
 Masaaki Sugihara
 Hui Sun
 Ágnes Szántó
 Akira Terui
 Thorsten Theobald
 Emmanuel Thomé
 Ashish Tiwari
 Maria-Laura Torrente
 Philippe Trébuchet
 Elias Tsigaridas
 William J. Turner
 Róbert Vajda
 Xiaoshen Wang
 Jacques-Arthur Weil
 Volker Weispfenning
 Thomas Wolf
 Bican Xia
 Zhengfeng Yang
 Chee Yap
 Liang Ye
 Alberto Zanoni
 Doron Zeilberger
 Zhonggang Zeng
 Mingbo Zhang
 Yang Zhang
 Jun Zhao
 Eugene Zima

ISSAC 2010



is organized by



Gesellschaft für Informatik



Fachgruppe Computeralgebra



Technische Universität München

Technische Universität München

in cooperation with



Association for
Computing Machinery

Advancing Computing as a Science & Profession

Association for Computing Machinery



Special Interest Group in Symbolic
and Algebraic Manipulation

supported by



Deutsche Forschungsgemeinschaft

sponsored by



Maplesoft, Waterloo, Canada

Algebraic Invariants and Their Differential Algebras

Evelyne Hubert
INRIA Méditerranée
Sophia Antipolis, France
Evelyne.Hubert@inria.fr

ABSTRACT

We review the algebraic foundations we developed to work with differential invariants of finite dimensional group actions. Those support the algorithms we introduced to operate symmetry reduction with a view towards differential elimination.

Categories and Subject Descriptors

I.1.4 [Computing Methodologies]: Symbolic and Algebraic Manipulation Applications; J.2 [Computer Applications]: Physical Sciences and Engineering Mathematics and statistics [Differential Geometry]; I.4.7.1 [Computing Methodologies]: Image Processing and Computer Vision Feature Measurement [Invariants]

General Terms

Theory

Keywords

Symmetry, Algebraic invariants, Differential Invariants, Moving frame, Differential Algebra, Differential Elimination

1. MOTIVATION AND BACKGROUND

A great variety of group actions arise in mathematics, physics, science and engineering and their invariants, whether algebraic or differential, are commonly used for classification and solving equivalence problems, or symmetry reduction. Equivalence problems essentially relies on determining *separating* invariants, i.e. functions whose values distinguish the orbits from one another. Symmetry reduction postulates that invariants are the best coordinates in which to think a problem. Here a *generating* set of invariants is needed, so as to rewrite the problem in terms of those. For any further computational use one also requires to know the syzygies, that is, relationships among those generating invariants.

Contrary to the algebraic theory of invariants, the theoretical support for the differential invariants we concentrate on belongs to differential geometry and analysis. There, the

constructions are often local and the use of the implicit function theorem is not considered a problem. To bring those ideas to algorithms there is first a need for firm algebraic foundations. Those algebraic foundations are what we want to review in this talk, covering the content of [8, 12, 13, 14, 10, 11].

On one hand, the restricted question of the finite generation of differential invariants was addressed by [31, 18, 19, 20, 25], in the more general case of pseudo-groups - see also [29, 26] for Lie groups.

In differential geometry equivalence problems are diverse though their resolutions often take their roots in the work of Elie Cartan [2]. Separating invariants are exhibited by a normalization procedure within the structure equations on the Maurer-Cartan forms [3, 5, 6, 15, 16] We shall call them the *Maurer-Cartan invariants*.

In their reinterpretation of Cartan's moving frame, Fels and Olver [4] addressed equivalence problems as well as finite generation, with applications beyond geometry [27, 22]. *Normalized invariants*, which are obtained as the normalization of the coordinate function on the space, are the focus there.

Our initial motivation resided in the symmetry reduction with a view towards differential elimination. The pioneering work of E. Mansfield [21] proved the adequacy of the tools provided by the reinterpretation of the moving frame method by M. Fels and P. Olver [4] and yet left many open problems.

2. PRESENTATION

We characterize three different sets of generating differential invariants together with their syzygies and rewriting algorithms. For the *normalized invariants* the syzygies can be constructed based on the recurrence formulae [10]. For the *edge invariants*, introduced in [28] and generalized in [10], the syzygies can then be deduced by differential elimination [8]. It is then a rather simple and yet significant observation we made that the geometry meaningful Maurer-Cartan invariants be generating [11]. This is of importance in the subject of evolution of curves and how it relates to integrable systems [1, 23, 24]. Their syzygies naturally emerge from the structure equations of the group [11].

The algebraic characterization of normalized invariants [13] entails an algorithm to compute and manipulate them explicitly [12]. We can then determine the other generating sets explicitly. But this is not needed for the rewriting.

The rewriting algorithms are all based on the trivial rewriting for the normalized invariants together with the relationship between *invariant derivations* and *invariantization*, two

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

key concepts introduced in [4] and revisited in [13, 10]. Beside their geometrical significance, the use of Maurer-Cartan invariants at that stage avoids the introduction of denominators.

We formalize the notion of syzygies through the introduction of the algebra of *monotone derivatives*. Along the lines of [8], this algebra is equipped with derivations that are defined inductively so as to encode their nontrivial commutation rules. The type of differential algebra introduced at this stage was shown to be a natural generalization of classical differential algebra [30, 17]. In the polynomial case, it is indeed endowed with an effective differential elimination theory that has been implemented [7, 8].

Let us point out that, while computing explicitly some (differential) invariants requires the knowledge of the action, only the knowledge of the infinitesimal generators of the action is needed for the determination of a generating set, the rewriting in terms of those and construction of the syzygies. Provided the action is given by rational functions, which covers most of encountered cases in the literature, the whole approach is algorithmic and implemented [9].

There is nonetheless one more input that is needed: a choice of cross-section. Though this can be chosen with a lot of freedom, in theory, practice shows that some are better than others. Strategy or theoretical understanding for the choice of the proper cross-section is an open problem.

3. REFERENCES

- [1] A. Calini, T. Ivey, and G. Marí-Beffa. Remarks on KdV-type flows on star-shaped curves. *Phys. D*, 238(8):788–797, 2009.
- [2] E. Cartan. *La méthode du repère mobile, la théorie des groupes continus, et les espaces généralisés*, volume 5 of *Exposés de Géométrie*. Hermann, Paris, 1935.
- [3] J. Clelland. Lecture notes from the MSRI workshop on Lie groups and the method of moving frames. <http://math.colorado.edu/~jnc/MSRI.html>, 1999.
- [4] M. Fels and P. J. Olver. Moving coframes. II. Regularization and theoretical foundations. *Acta Appl. Math.*, 55(2):127–208, 1999.
- [5] R. B. Gardner. The method of equivalence and its applications. *SIAM, Philadelphia*, 1989.
- [6] P. A. Griffiths. On Cartan’s method of Lie groups as applied to uniqueness and existence questions in differential geometry. *Duke Math. J.*, 41:775–814, 1974.
- [7] E. Hubert. DIFFALG: *extension to non commuting derivations*. INRIA, Sophia Antipolis, 2005. www.inria.fr/cafe/Evelyne.Hubert/diffalg.
- [8] E. Hubert. Differential algebra for derivations with nontrivial commutation rules. *Journal of Pure and Applied Algebra*, 200(1-2):163–190, 2005.
- [9] E. Hubert. *The MAPLE package AIDA - Algebraic Invariants and their Differential Algebras*. INRIA, 2007. <http://www.inria.fr/cafe/Evelyne.Hubert/aida>.
- [10] E. Hubert. Differential invariants of a Lie group action: syzygies on a generating set. *Journal of Symbolic Computation*, 44(3):382–416, 2009.
- [11] E. Hubert. Generation properties of Maurer-Cartan invariants. Preprint <http://hal.inria.fr/inria-00194528>, 2010.
- [12] E. Hubert and I. A. Kogan. Rational invariants of a group action. Construction and rewriting. *Journal of Symbolic Computation*, 42(1-2):203–217, 2007.
- [13] E. Hubert and I. A. Kogan. Smooth and algebraic invariants of a group action. Local and global constructions. *Foundations of Computational Mathematics*, 7(4), 2007.
- [14] E. Hubert and P. J. Olver. Differential invariants of conformal and projective surfaces. *Symmetry Integrability and Geometry: Methods and Applications*, 3(097), 2007.
- [15] T. A. Ivey and J. M. Landsberg. *Cartan for beginners: differential geometry via moving frames and exterior differential systems*, volume 61 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2003.
- [16] G. R. Jensen. *Higher order contact of submanifolds of homogeneous spaces*. Springer-Verlag, Berlin, 1977. Lecture Notes in Mathematics, Vol. 610.
- [17] E. R. Kolchin. *Differential Algebra and Algebraic Groups*, volume 54 of *Pure and Applied Mathematics*. Academic Press, 1973.
- [18] A. Kumpera. Invariants différentiels d’un pseudogroupe de Lie. In *Lecture Notes in Math.*, Vol. 392. Springer, Berlin, 1974.
- [19] A. Kumpera. Invariants différentiels d’un pseudogroupe de Lie. I. *J. Differential Geometry*, 10(2):289–345, 1975.
- [20] A. Kumpera. Invariants différentiels d’un pseudogroupe de Lie. II. *J. Differential Geometry*, 10(3):347–416, 1975.
- [21] E. L. Mansfield. Algorithms for symmetric differential systems. *Foundations of Computational Mathematics*, 1(4):335–383, 2001.
- [22] E. L. Mansfield. *A practical guide to the invariant calculus*. Number 26 in Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2010.
- [23] G. Marí Beffa. Projective-type differential invariants and geometric curve evolutions of KdV-type in flat homogeneous manifolds. *Annales de l’Institut Fourier (Grenoble)*, 58(4):1295–1335, 2008.
- [24] G. Marí Beffa. Hamiltonian evolution of curves in classical affine geometries. *Physica D. Nonlinear Phenomena*, 238(1):100–115, 2009.
- [25] J. Muñoz, F. J. Muriel, and J. Rodríguez. On the finiteness of differential invariants. *J. Math. Anal. Appl.*, 284(1):266–282, 2003.
- [26] P. J. Olver. *Equivalence, Invariants and Symmetry*. Cambridge University Press, 1995.
- [27] P. J. Olver. Moving frames: a brief survey. In *Symmetry and perturbation theory (Cala Gonone, 2001)*, pages 143–150. World Sci. Publishing 2001.
- [28] P. J. Olver. Generating differential invariants. *Journal of Mathematical Analysis and Applications*, 333:450–471, 2007.
- [29] L. V. Ovsianikov. *Group analysis of differential equations*. Academic Press Inc., New York, 1982.
- [30] J. F. Ritt. *Differential Algebra*, volume XXXIII of *Colloquium publications*. American Mathematical Society, 1950. http://www.ams.org/online_bks.
- [31] A. Tresse. Sur les invariants des groupes continus de transformations. *Acta Mathematica*, 18:1–88, 1894.

Verification Methods : Rigorous Results using Floating-Point Arithmetic

Siegfried M. Rump
Institute for Reliable Computing
Hamburg University of Technology
Schwarzenbergstraße 95, 21071 Hamburg, Germany
and
Visiting Professor at Waseda University
Faculty of Science and Engineering
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
rump@tu-harburg.de

Categories and Subject Descriptors

F2.2.1 [Numerical Algorithms and Problems]

General Terms

Verification

ABSTRACT

The classical mathematical proof is performed by pencil and paper. However, there are many ways in which computers may be used in a mathematical proof. But “proofs by computers” or even the use of computers in the course of a proof are not so readily accepted (the December 2008 issue of the Notices of the American Mathematical Society is devoted to formal proofs by computers).

In this talk we discuss how *verification methods* may assist in achieving a mathematically rigorous result. In particular we emphasize how floating-point arithmetic is used.

The goal of verification methods is ambitious: For a given problem it is proved, with the aid of a computer, that there exists a (unique) solution within computed bounds. The methods are constructive, and the results are rigorous in every respect. Verification methods apply to data with tolerances as well.

Rigorous results are the main goal in computer algebra. However, verification methods use solely floating-point arithmetic, so that the total computational effort is not too far from that of a purely (approximate) numerical method. Non-trivial problems have been solved using verification methods. For example:

Tucker (1999) received the 2004 EMS prize awarded by the European Mathematical Society for (citation) “giving a rigorous proof that the Lorenz attractor exists for the parameter values provided by Lorenz. This was a long stand-

ing challenge to the dynamical system community, and was included by Smale in his list of problems for the new millennium. The proof uses computer estimates with rigorous bounds based on higher dimensional interval arithmetics.”

Sahinidis and Tawaralani (2005) received the 2006 Beale-Orchard-Hays Prize for their package BARON which (citation) “incorporates techniques from automatic differentiation, interval arithmetic, and other areas to yield an automatic, modular, and relatively efficient solver for the very difficult area of global optimization”.

A main goal of this talk is to introduce the principles of how to design verification algorithms, and how these principles differ from those for traditional numerical algorithms.

We begin with a brief discussion of the working tools of verification methods, in particular floating-point and interval arithmetic. In particular the development and limits of verification methods for finite dimensional problems are discussed in some detail; problems include dense systems of linear equations, sparse linear systems, systems of nonlinear equations, semi-definite programming and other special linear and nonlinear problems including M-matrices, simple and multiple roots of polynomials, bounds for simple and multiple eigenvalues or clusters, and quadrature. We mention that automatic differentiation tools to compute the range of gradients, Hessians, Taylor coefficients, and slopes are necessary. If time permits, verification methods for continuous problems, namely two-point boundary value problems and semilinear elliptic boundary value problems are presented.

Throughout the talk, a number of examples of the *wrong* use of interval operations are given. In the past such examples contributed to the dubious reputation of interval arithmetic, whereas they are, in fact, just a misuse.

Some algorithms are presented in executable Matlab/INTLAB-code. INTLAB, the Matlab toolbox for reliable computing and free for academic use, is developed and written by Rump (1999). It was, for example, used by Bornemann, Laurie, Wagon, and Waldvogel (2004) in the solution of half of the problems of the 10×10 -digit challenge by Trefethen (2002).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

References

- F. Bornemann, D. Laurie, S. Wagon, and J. Waldvogel. *The SIAM 100-Digit Challenge—A Study in High-Accuracy Numerical Computing*. SIAM, Philadelphia, 2004.
- S.M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999a. URL <http://www.ti3.tu-harburg.de/rump/intlab/index.html>.
- N.V. Sahinidis and M. Tawaralani. A polyhedral branch-and-cut approach to global optimization. *Math. Programming*, B103:225–249, 2005.
- L. N. Trefethen. The SIAM 100-Dollar, 100-Digit Challenge. *SIAM-NEWS*, 35(6):2, 2002. <http://www.siam.org/siamnews/06-02/challengedigits.pdf>.
- W. Tucker. The Lorenz attractor exists. *C. R. Acad. Sci., Paris, Sér. I, Math.*, 328(12):1197–1202, 1999.

Theory of Reals for Verification and Synthesis of Hybrid Dynamical Systems

Ashish Tiwari^{*}
SRI International
333 Ravenswood Ave.
Menlo Park, CA
tiwari@csl.sri.com

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—*Algebraic algorithms*; I.2.3 [Artificial Intelligence]: Deduction and Theorem Proving—*Inference engines*

General Terms

Verification, Algorithms

Keywords

Hybrid Dynamical Systems, Theory of Reals

ABSTRACT

Real numbers are used to model all physical processes around us. The temperature of a room, the speed of a car, the angle of attack of an airplane, protein concentration in a cell, blood glucose concentration in a human, and the amount of chemical in a tank are a few of the countless quantities that arise in science and engineering and that are modeled using real-valued variables. Many of these physical quantities are now being *controlled* by embedded software running on some hardware platform. The resulting systems could possibly be communicating and coordinating with other such systems and reacting actively to changes in their environment. The net result is a complex cyber-physical system. Several such systems operate in safety-critical domains, such as transportation and health care, where failures can potentially cause a lot of financial as well as human life loss. Formal verification and synthesis are both indispensable components of any methodology for designing and efficiently developing safe cyber-physical systems.

Mathematically, a cyber-physical system is a dynamical system – a system that evolves or changes over time. Dynamical systems are modeled using many different formalisms, such as, discrete state transition systems, continuous-time

dynamical systems, and hybrid systems. Hybrid dynamical systems is a popular formalism for modeling a large class of complex systems, including control systems, embedded systems, robotic systems, and biological systems. There are several different approaches for performing verification of a hybrid dynamical system, but almost all approaches rely on algorithms for reasoning in the *theory of reals*.

Tarski [5] showed that the first-order theory of the reals is decidable [5]. However, Tarski's procedure has hyperexponential complexity, and hence, a more efficient algorithm, based on “cylindrical algebraic decomposition” (CAD), was proposed by Collins [1]. The original CAD algorithm was further refined and optimized over the years by several authors, and it has been implemented in an impressive tool called QEPCAD [2]. It has also been argued that, for the theory of reals, algorithms with better theoretical complexity may not necessarily perform better in practice. Since the problem of reasoning on nonlinear constraints is inherently hard, there is a need for discovering and implementing procedures that best match the requirements of a particular application.

The verification application alluded to above generates nonlinear constraints of mainly two different forms: $\forall \vec{x} : \phi(\vec{x})$ and $\exists \vec{a} : \forall \vec{u} : \phi(\vec{a}, \vec{u})$, where $\vec{x}$ and $\vec{a}$ denote a sequence of variables and $\phi(\vec{x})$ denotes a formula in the theory of reals over the variables $\vec{x}$. The number of variables can be large and a procedure should be able to handle at least 20 to 30 variables to be useful. Another requirement is that the procedure should not fail and should always terminate. While not a strict requirement, incrementality in handling of the constraints is a useful attribute in verification applications. However, our application also provides some *flexibilities* that can be favorably used to design more practical procedures for reasoning over nonlinear constraints. A crucial flexibility provided by the verification application is that it is “incompleteness-tolerant”, that is, it can still use incomplete, but sound, procedures. Even though the formulas ϕ are invariably large, not all parts of the formula are relevant for proving (or disproving) its validity.

Recently, we have proposed a procedure for detecting unsatisfiability of a conjunction of polynomial equations and inequalities that is favorably suited for our particular application. The procedure is based on Gröbner basis computation. It can be seen as a generalization of the Simplex procedure for linear arithmetic to the case of nonlinear arithmetic. The procedure works in two steps. In the first step, all inequalities are converted into equations by introducing new variables. After applying the first step, we get a con-

^{*}Supported in part by NSF grants CNS-0720721 and CSR-0917398 and NASA grant NNX08AB95A.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

junction of equations (and some disequations of the form $u \neq 0$, where u is a new variable) and we have to decide its satisfiability. Let us denote the new set of equations as $P = 0$, where P is a set of polynomials. In the second step, we search for a *positive definite polynomial* in the ideal of P . If we successfully find a positive definite polynomial in the ideal of P , then we can conclude that the original formula was unsatisfiable.

For finding a positive definite polynomial in an ideal, we use Gröbner basis computation under different term orderings. The observation here is that members of an ideal that are minimal (in the ordering) appear explicitly in the Gröbner basis (constructed using that ordering). The refutational completeness of the procedure is a consequence of the Positivstellensatz theorem from real algebraic geometry [6].

There are several other efforts in building practical procedures for reasoning with nonlinear constraints. One popular approach is based on detecting if a polynomial is a *sum-of-squares* (positive definite) using semi-definite programming [4, 3]. This suggests the natural idea of combining procedures based on numerical calculations and symbolic reasoning to obtain practical and scalable solvers for nonlinear arithmetic.

1. REFERENCES

- [1] G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Proc. 2nd GI Conf. Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.
- [2] H. Hong and C. Brown. Quantifier elimination procedure by cylindrical algebraic decomposition. www.usna.edu/Users/cs/qepcad/B/QEPCAD.html.
- [3] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming Ser. B*, 96(2):293–320, 2003.
- [4] S. Prajna, A. Papachristodoulou, and P. A. Parrilo. SOSTOOLS: Sum of Square Optimization Toolbox, 2002. <http://www.cds.caltech.edu/sostools>.
- [5] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1948. Second edition.
- [6] A. Tiwari. An algebraic approach for the unsatisfiability of nonlinear constraints. In *Computer Science Logic, 14th Annual Conf., CSL 2005*, volume 3634 of *LNCS*, pages 248–262. Springer, 2005.

Symbolic Methods for Solving Systems of Linear Ordinary Differential Equations

Moulay A. Barkatou
University of Limoges; CNRS
XLIM/DMI UMR 6172
87060 Limoges, France
moulay.barkatou@unilim.fr

ABSTRACT

The main purpose of this tutorial is to present and explain symbolic methods for studying systems of linear ordinary differential equations with emphasis on direct methods and their implementation in computer algebra systems.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms

General Terms

Algorithms

Keywords

Computer Algebra, Linear Systems of Differential Equations, Reduction Algorithms, Singularities, Formal Solutions, Closed-form Solutions

1. INTRODUCTION

Whether one is interested in global problems (finding closed form solutions, testing reducibility, computing properties of the differential Galois group) or in local problems (computing formal invariants or formal solutions) of linear differential scalar equations or systems, one has to develop and use appropriate tools for local analysis the purpose of which is to describe the behavior of the solutions near a point x_0 without knowing these solutions in advance.

After introducing the basic tools of local analysis we present the state of the art of existing algorithms and programs for solving the main local problems such as determining the type of a given singularity, computing the rank of a singularity, computing the Newton polygon and Newton polynomials at a given singularity, finding formal solutions, etc. Next we explain how by piecing together the local information around the different singularities one can solve some global problems such as finding rational solutions, exponential solutions, fac-

toring a given differential system. The last part of the tutorial will present some recent developments including algorithms for solving directly systems of higher order differential equations, algorithms for differential systems in positive characteristic and formal reduction of pfaffian systems.

2. REFERENCES

- [1] S. A. Abramov, M. Bronstein, D. E. Khmelnov. On Regular and Logarithmic Solutions of Ordinary Linear Differential Systems. CASC 2005: 1-12.
- [2] D.G. Babbitt, V.S. Varadarajan. Formal reduction of meromorphic differential equations: a group theoretic view. Pacific Journal of Mathematics, 109(1), 1-80, 1983.
- [3] W. Balser. Formal power series and linear systems of meromorphic ordinary differential equations, Springer-Verlag (2000).
- [4] W. Balser, W. B. Jurkat, D. A. Lutz. A General Theory of Invariants for Meromorphic Differential Equations; Part I, Formal Invariants, Funkcialaj ekvacioj 22, (1979), 197-221.
- [5] M. A. Barkatou. An algorithm for computing a companion block diagonal form for a system of linear differential equations, AAEC 4 (1993), 185-195.
- [6] M. A. Barkatou. A Rational Version of Moser's Algorithm. ISSAC 1995: 297-302.
- [7] M. A. Barkatou. An algorithm to compute the exponential part of a formal fundamental matrix solution of a linear differential system, AAEC, 8-1 (1997) 1-23.
- [8] M. A. Barkatou. On Rational Solutions of Systems of Linear Differential Equations. Journal of Symbolic Computation 28, 547-567, 1999.
- [9] M. A. Barkatou. On super-irreducible forms of linear differential systems with rational function coefficients. Journal of Computational and Applied Mathematics, 162(1):1-15, 2004.
- [10] M. A. Barkatou. Factoring systems of linear functional equations using eigenrings. I. S. Kotsireas and E. V. Zima, editors, Latest Advances in Symbolic Algorithms, pages 22-42. World Scientific, 2007.
- [11] M. A. Barkatou, E. Pflügel. An algorithm computing the regular formal solutions of a system of linear differential equations. Journal of Symbolic Computation, 28:569-588, 1999.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25-28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

- [12] M. A. Barkatou, N. Le Roux. Rank reduction of a class of pfaffian systems in two variables. ISSAC 2006: 204-211.
- [13] M. A. Barkatou, E. Pflügel. On the Moser- and super-reduction algorithms of systems of linear differential equations and their complexity. J. Symbolic Computation, 44(8):1017-1036, 2009.
- [14] M. A. Barkatou, E. Pflügel. Computing super-irreducible forms of systems of linear differential equations via moser-reduction: a new approach. ISSAC 2007: 1-8.
- [15] M. A. Barkatou, E. Pflügel. On the Equivalence Problem of Linear Differential Systems and Its Application for Factoring Completely Reducible Systems. ISSAC 1998: 268-275.
- [16] M. A. Barkatou, E. Pflügel. The ISOLDE package, a SourceForge Open Source project, <http://isolde.sourceforge.net>.
- [17] M. A. Barkatou, T. Cluzeau, C. El Bacha. Algorithms for regular solutions of higher-order linear differential systems. ISSAC 2009: 7-14.
- [18] M. A. Barkatou, T. Cluzeau, J.-A. Weil. Factoring partial differential systems in positive characteristic. Differential Equations and Symbolic Computations (DESC) Book, Editor D. Wang, Birkhäuser, 2005.
- [19] T. Cluzeau. Factorization of differential systems in characteristic p . In: Proceedings of ISSAC'03. ACM, New York, NY, USA, pp. 58-65.
- [20] T. Cluzeau. Algorithmique modulaire des équations différentielles linéaires. Thèse de l'université de Limoges, Septembre 2004.
- [21] G. Chen. An algorithm for computing the formal solutions of differential systems in the neighborhood of an irregular singular point. ISSAC'90, pp. 231-235.
- [22] R. C. Churchill, J. J. Kovacic. Cyclic vectors, Differential Algebra and related topics. World Sci. Publishing (2002), 191-218.
- [23] F. T. Cope. Formal solutions of irregular differential equations, Part II, Amer. J. Math. 58 (1936), 130-140.
- [24] E. A. Coddington, N. Levinson. Theory of Ordinary Differential Equations, Mc Graw-Hill Book Company, INC New York (1955)
- [25] E. Corel. Algorithmic computation of exponents for linear differential systems. From combinatorics to dynamical systems, 17-61, IRMA Lect. Math. Theor. Phys., 3, de Gruyter, Berlin, 2003.
- [26] V. Dietrich. Zur Reduktion von linearen Differentialgleichungssystemen. Math. Ann. 237, (1978) 79-95.
- [27] A. Hilali, A. Wazner. Formes super-irréductibles des systèmes différentiels linéaires. Numerische Mathematik, 50:429-449, 1987.
- [28] A. H. M. Levelt. Stabilizing differential operators: a method for computing invariants at irregular singularities, CADE (1991) Comput. Math. and Appl., M. Singer Editor, Academic Press Ltd, 181-128.
- [29] M. Loday-Richaud. Solutions formelles des systèmes différentiels linéaires méromorphes et sommation. Exposition. Math. 13 (1995), no. 2-3, 116-162.
- [30] D. A. Lutz, R. Schäfke. On the identification and stability of formal invariants for singular differential equations. Linear Algebra and its Applications 72, (1985) 1-46.
- [31] J. Moser. The order of a singularity in Fuchs' theory, Math. Z. 72, (1960), 379-398.
- [32] E. Pflügel. An Algorithm for Computing Exponential Solutions of First Order Linear Differential Systems. ISSAC 1997: 164-17.
- [33] E. Pflügel. Effective formal reduction of linear differential systems. Appl. Alg. Eng. Comm. Comp. 10 (2), (2000) 153-187.
- [34] M. van der Put and M. F. Singer. Galois Theory of Linear Differential Equations Grundlehren der mathematischen Wissenschaften, vol. 328, Springer, 2003.
- [35] R. Sommeling. Characteristic classes for irregular singularities. PhD thesis, University of Nijmegen, 1993.
- [36] L. Sauvage. Sur les solutions régulières d'un système d'équations différentielles, Ann. Sc. ENS 3, (1886), 391-404.
- [37] Y. Sibuya, Linear differential equations in the complex domain: problems of analytic continuation, Translated from Japanese by the author, Translations of Mathematical Monographs 82, A.M.S. (1990).
- [38] H. L. Turrittin. Convergent solutions for ordinary linear homogeneous differential equations in the neighborhood of an irregular singular point, Acta Math. 93, (1955), 27-66
- [39] V. S. Varadarajan. Linear meromorphic differential equations: a modern point of view. Bull. Amer. Math. Soc. (N.S.) 33 (1996), no. 1, 1-42.
- [40] E. Wagenführer. On the Computation of Formal Fundamental Solutions of Linear Differential Equations at a Singular Point. Analysis, 9: 389-405, 1989.
- [41] E. Wagenführer. Formal series solutions of singular systems of linear differential equations and singular matrix pencils. J. Fac. Sci. Univ. Tokyo Sect. IA Math. 36 (1989), no. 3, 681-702.
- [42] W. Wasow. Asymptotic expansions for ordinary differential equations, Interscience, New York, (1965); reprint R.E. Krieger Publishing Co, inc. (1976).

Asymptotically Fast Algorithms for Modern Computer Algebra

Jürgen Gerhard
Maplesoft
Waterloo, Ontario, Canada
jgerhard@maplesoft.com

ABSTRACT

The solution of computational tasks from the “real world” requires high performance computations. Not limited to mathematical computing, asymptotically fast algorithms have become one of the major contributing factors in this area. Based on [4], the tutorial will give an introduction to the beauty and elegance of modern computer algebra.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms; F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—*computations on polynomials*; G.4 [Mathematics of Computing]: Mathematical Software—*algorithm design and analysis, efficiency*

General Terms

Theory

Keywords

fast algorithms, computer algebra

1. INTRODUCTION

Mathematical software has long advanced beyond its academic nursery and has become a standard tool in industrial design and analysis practice. The size of industrial problems made it necessary not just for numerical computation packages but also for computer algebra systems to increase their performance. More and more symbolic computation systems and packages, such as GMP [14], Magma [2], Maple [7], and NTL [10], now feature implementations of asymptotically fast algorithms that scale almost linearly with the input size.

2. FAST ALGORITHMS

At the heart of asymptotically fast polynomial arithmetic lies fast multiplication, e.g., by the fast Fourier transform [3].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

An abundance of higher-level computational problems for polynomials, including but not limited to division [11], evaluation and interpolation [1], greatest common divisors [13], factorization [15], symbolic integration [5] and symbolic summation [6] can be reduced to polynomial multiplication, and any speedup in the underlying basic polynomial arithmetic immediately translates into a speedup of about the same order of magnitude for these advanced algorithms as well.

The techniques for fast symbolic computation have some of their roots in numerical analysis (fast Fourier transform, Newton iteration) and computer science (e.g., divide-and-conquer, work balancing). A somewhat unique ingredient in computer algebra, however, is the omnipresent and powerful scheme of modular algorithms.

The techniques mentioned above work well not only for polynomial arithmetic, but can be extended (with some notable exceptions) to integer arithmetic as well [9], and often the same or similar techniques apply or can be used in linear algebra [12].

Optimizations make asymptotically fast algorithms practical and powerful. Determining the break-even points between classical and fast algorithms for hybrid schemes can be challenging and platform dependent. The Golden Rules of Schönhage et al [8] for the development of fast algorithms apply, notably:

- Do care about the size of $O(1)$!
- Do not waste a factor of two!
- Don’t forget the algorithms in object design!
- Clean results by approximate methods is sometimes much faster!

3. REFERENCES

- [1] A. Borodin and I. Munro. *The Computational Complexity of Algebraic and Numeric Problems*, volume 1 of *Theory of computation series*. American Elsevier Publishing Company, New York, 1975.
- [2] J. J. Cannon and W. Bosma, editors. *Handbook of Magma Functions, Edition 2.13*. 2006. magma.maths.usyd.edu.au.
- [3] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.
- [4] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2nd edition, 2003.

- [5] J. Gerhard. Fast modular algorithms for squarefree factorization and Hermite integration. *Applicable Algebra in Engineering, Communication and Computing*, 11(3):203–226, 2001.
- [6] J. Gerhard. *Modular Algorithms in Symbolic Summation and Symbolic Integration*, volume 3218 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.
- [7] Maplesoft. Maple – Math & Engineering Software. www.maplesoft.com/products/Maple/index.aspx.
- [8] A. Schönhage, A. F. W. Grotefeld, and E. Vetter. *Fast Algorithms – A Multitape Turing Machine Implementation*. BI Wissenschaftsverlag, Mannheim, 1994.
- [9] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.
- [10] V. Shoup. NTL: A library for doing number theory. www.shoup.net/ntl.
- [11] M. Sieveking. An algorithm for division of powerseries. *Computing*, 10:153–156, 1972.
- [12] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.
- [13] V. Strassen. The computational complexity of continued fractions. *SIAM Journal on Computing*, 12(1):1–27, 1983.
- [14] Various authors. The GNU multiple precision arithmetic library. www.gmp1ib.org.
- [15] H. Zassenhaus. On Hensel factorization, I. *Journal of Number Theory*, 1:291–311, 1969.

Transformation and Factorization of Partial Differential Systems: Applications to Stochastic Systems

S.P. Tsarev
Siberian Federal University
Svobodnyi avenue, 79
660041, Krasnoyarsk, Russia
sptsarev@mail.ru

ABSTRACT

Factorization of linear ordinary differential operators with variable coefficients is one of the well-known methods used for solution of the corresponding differential equations. Factorization of linear partial differential operators is a much more complicated problem. An appropriate modification of the “naive” definition of factorization gives rise to a bunch of new methods of closed-form solution of a single linear partial differential equation or a system of such equations. This approach is naturally related to differential transformations—another popular method of solution of differential equations.

Categories and Subject Descriptors

G.0 [Mathematics of Computing]: GENERAL

General Terms

Theory

Keywords

Integration of partial differential equations, factorization

1. INTRODUCTION

Factorization of polynomials is a popular topic in lecture courses presenting modern algorithms of computer algebra. Factorization of linear ordinary differential operators with variable coefficients (LODOs) is less known but is an important method used for solution of the corresponding differential equations. Now we have a complete (although highly complex) algorithm for factorization of an arbitrary LODO with rational functions as coefficients. A number of important old [5, 6] and new results in this field are summarized in [13]. Theory of factorization of linear partial differential operators (LPDOs) is even less popular due to a simple fact: a “naive” definition of factorization of a given LPDO L as its representation as a composition $L = L_1 \circ L_2$ of lower-order

operators does not enjoy good algebraic properties and in general is not related to existence of a complete closed-form solution.

In this tutorial we present the old but still fruitful Laplace cascade method for solution of a class of linear partial differential equations as well as a number of its latest modifications and generalizations. We explain an important and nontrivial link to a generalized notion of factorization of LDPOs proposed recently [9, 11]. An application [3] to solution of an interesting system of linear partial differential equations describing the behavior of a simple nonlinear stochastic ordinary differential equation will be described.

2. ALGEBRAIC THEORIES OF FACTORIZATION FOR LODOs AND LPDOs

One of the basic algebraic results in the theory of factorization of LODOs was obtained in the beginning of the 20th century:

THEOREM 1 (E. LANDAU [5]). *Any two different decompositions of a given LODO L into products of irreducible LODOs $L = P_1 \cdots P_k = \bar{P}_1 \cdots \bar{P}_p$ have the same number of factors ($k = p$) and the factors have equal orders (after a transposition).*

Unfortunately simple examples (see e.g. [9]) show that for LPDOs this nice theorem does not hold. An algebraic explanation lies in the fact that the (noncommutative) ring of LODOs has only principal left and right ideals, i.e. every its ideal is generated by one LODO; for the ring of LPDOs this is no longer true—in fact already the ring of commutative multivariate polynomials does not have this property, although the latter is a unique factorization domain. This implies that one should develop an alternative algebraic definition of factorization capable of providing an analogue of Landau theorem and integrating into a unified framework different methods of reduction of the order of linear partial differential equations with various classical transformations, such as Laplace and Moutard transformations (see [2]).

A number of partial algorithms for closed-form solution of systems of linear partial differential equations are known, starting with the almost forgotten old results [7, 8]. These and different modern results [1, 4, 10, 12] may serve as a hint for existence of a unified view.

An algebraic approach to the problem was proposed in [9, 11]. An adequate algebraic structure suitable to serve as a basis for unification of all known partial results will be exposed.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

3. ACKNOWLEDGMENTS

This research was partially supported by the RFBR Grant 09-01-00762-a.

4. REFERENCES

- [1] C. Athorne. A $\mathbf{Z}^2 \times \mathbf{R}^3$ Toda system. *Phys. Lett. A*, 206:162–166, 1995.
- [2] G. Darboux. *Leçons sur la théorie générale des surfaces et les applications géométriques du calcul infinitésimal*. T. 2. Gauthier-Villars, 1889.
- [3] E.I. Ganzha, V.M. Loginov, S.P. Tsarev. Exact solutions of hyperbolic systems of kinetic equations. Application to Verhulst model with random perturbation. *Mathematics of Computation*, 1(3):459–472, 2008. e-print <http://www.arxiv.org/math.AP/0612793>.
- [4] D. Grigoriev, F. Schwarz. Factoring and solving linear partial differential equations, *Computing*, 73:179–197, 2004.
- [5] E. Landau. Über irreduzible Differentialgleichungen. *J. für die reine und angewandte Mathematik* 124:115–120, 1902.
- [6] A. Loewy. Über reduzible lineare homogene Differentialgleichungen, *Math. Annalen* 56:549–584, 1903.
- [7] J.Le Roux. Extensions de la méthode de Laplace aux équations linéaires aux dérivées partielles d'ordre supérieur au second. *Bull. Soc. Math. France*, 27:237–262, 1899. A digitized copy is obtainable from <http://www.numdam.org/>
- [8] L. Petré. Extension de la méthode de Laplace aux équations $\sum_{i=0}^{n-1} A_{1i} \frac{\partial^{i+1} z}{\partial x \partial y^i} + \sum_{i=0}^n A_{0i} \frac{\partial^i z}{\partial y^i} = 0$. *Lund Univ. Arsskrift*, 7(3):1–166, 1911.
- [9] S.P. Tsarev. Factorization of linear partial differential operators and Darboux integrability of nonlinear PDEs. *SIGSAM Bulletin* 32(4):21–28, 1998. E-print <http://www.arxiv.org/cs.SC/9811002>.
- [10] S.P. Tsarev. Generalized Laplace Transformations and Integration of Hyperbolic Systems of Linear Partial Differential Equations. *Proc. ISSAC'2005*, ACM Press. 2005. P. 325–331; also e-print cs.SC/0501030 at <http://www.archiv.org/>.
- [11] S.P. Tsarev. Factorization of Linear Differential Operators and Systems. In: *Algebraic Theory of Differential Equations* (Eds. M. A. H. MacCallum, A. V. Mikhailov), London Mathematical Society Lecture Note Series (No. 357), CUP, 2008, p. 111–131. Also E-print: <http://arxiv.org/abs/0801.1341>
- [12] Ziming Li, F. Schwarz, S.P. Tsarev. Factoring systems of linear PDEs with finite-dimensional solution spaces. *J. Symbolic Computation* 36:443–471, 2003.
- [13] M. van der Put, M.F. Singer. *Galois Theory of Linear Differential Equations*, Grundlehren der Mathematischen Wissenschaften, v. 328, Springer, 2003.

A New Incremental Algorithm for Computing Groebner Bases

Shuhong Gao^{*†}
Dept. of Math. Sciences
Clemson University
Clemson, SC 29634-0975
sgao@clemson.edu

Yinhua Guan
Dept. of Math. Sciences
Clemson University
Clemson, SC 29634-0975
gguan@clemson.edu

Frank Volny IV
Dept. of Math. Sciences
Clemson University
Clemson, SC 29634-0975
fvolny@g.clemson.edu

ABSTRACT

In this paper, we present a new algorithm for computing Gröbner bases. Our algorithm is incremental in the same fashion as F5 and F5C. At a typical step, one is given a Gröbner basis G for an ideal $\mathbf{I}$ and any polynomial g , and it is desired to compute a Gröbner basis for the new ideal $\langle \mathbf{I}, g \rangle$, obtained from $\mathbf{I}$ by joining g . Let $(\mathbf{I} : g)$ denote the colon ideal of $\mathbf{I}$ divided by g . Our algorithm computes Gröbner bases for $\langle \mathbf{I}, g \rangle$ and $(\mathbf{I} : g)$ simultaneously. In previous algorithms, S-polynomials that reduce to zero are useless, in fact, F5 tries to avoid such reductions as much as possible. In our algorithm, however, these “useless” S-polynomials give elements in $(\mathbf{I} : g)$ and are useful in speeding up the subsequent computations. Computer experiments on some benchmark examples indicate that our algorithm is much more efficient (two to ten times faster) than F5 and F5C.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—*Algebraic Algorithms*; F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—*Computations on Polynomials*

General Terms

Algorithms

Keywords

Gröbner basis, Buchberger’s Algorithm, Colon ideal, F5 Algorithm

^{*}The three authors were partially supported by National Science Foundation under grants DMS-0302549 and CCF-0830481, and National Security Agency under grant H98230-08-1-0030.

[†]We would also like to thank the referees for their very helpful comments and suggestions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

1. INTRODUCTION

In Buchberger’s algorithm (1965, [1, 2, 3]), one has to reduce many “useless” S-polynomials (i.e. those that reduce to 0 via long division), and each reduction is time consuming. Faugère (1999, [9]) introduced a reduction method (F4) that can efficiently reduce many polynomials simultaneously; see also Joux and Vtse (2010, [11]) for a recent variant of F4. Lazard (1983, [13]) pointed out the connection between a Gröbner basis and linear algebra, that is, a Gröbner basis can be computed by Gauss elimination of Macaulay matrices (1902, [14]). This idea is implemented as XL type algorithms by Courtois et al. (2000, [5]), Ding et al. (2008, [7]), Mohammed et al. (2008–2009, [15, 16]), and Buchman et al. (2010, [4]). The linear algebra approach can be viewed as a fast reduction method. The main problem with these approaches is that the memory usage grows very quickly, and in practice the computation for even a small problem can not be done simply due to memory running out.

Faugère (2002, [10]) introduced the idea of signatures and rewriting rules that can detect many useless S-polynomials hence saving a significant amount of time that would be used in reducing them. By computer experiments, Faugère showed that his algorithm F5 is many times faster than previous algorithms. However, F5 seems difficult to both understand and implement. Eder and Perry (2009, [8]) simplified some of the steps in F5 and gave a variant called F5C which is almost always faster than F5. We should note that Sun and Wang (2009, [17]) also give a new proof and some improvement for F5.

Our main purpose of the current paper is to present a new algorithm that is both simpler and more efficient than F5 and F5C. Our algorithm is incremental just like F5 and F5C. Let $\mathbb{F}$ be any field and $R = \mathbb{F}[x_1, \dots, x_n]$. Fix an arbitrary monomial order on R . At a typical iterative step, a Gröbner basis G for an ideal $\mathbf{I}$ in R is already computed, and it is desired to compute a Gröbner basis for the new ideal $\langle \mathbf{I}, g \rangle$ for a given polynomial $g \in R$. In F5, the basis G may not be reduced, thus containing many redundant polynomials. F5C is the same as F5 except that G is replaced by a reduced Gröbner basis in the next iterative step. Our algorithm will use a reduced Gröbner basis G as in F5C, but the crucial difference is that we introduce a so-called “super top-reduction” to detect “useless” polynomials. Furthermore, if there happens to be a polynomial that reduces to 0, it will be used to detect more “useless” polynomials. Hence reduction to 0 in our algorithm is not “useless” at all. In fact, it gives

us a polynomial in the colon ideal

$$(\mathbf{I} : g) = \{u \in R : ug \in \mathbf{I}\}. \quad (1)$$

It is of independent interest to have an efficient algorithm for computing Gröbner bases for colon ideals of the form $(\mathbf{I} : g)$, as it is a routine repeatedly used in primary decomposition, especially in separating components of different dimensions.

In Section 2, we shall present a relation between the Gröbner bases of $(\mathbf{I}, g)$ and $(\mathbf{I} : g)$. This is based on the exact sequence of R -modules:

$$0 \longrightarrow R/(\mathbf{I} : g) \longrightarrow R/\mathbf{I} \longrightarrow R/(\mathbf{I}, g) \longrightarrow 0$$

where the second morphism is defined by multiplication by g , which is injective by the definition in (1), and the third is the canonical morphism. The exactness of the sequence implies that

$$\dim_{\mathbb{F}}(R/\mathbf{I}) = \dim_{\mathbb{F}}(R/(\mathbf{I}, g)) + \dim_{\mathbb{F}}(R/(\mathbf{I} : g)). \quad (2)$$

For an arbitrary ideal $\mathbf{I}$, we show in Section 2 how to compute $\mathbb{F}$ -linear bases for all of these vector spaces from a given Gröbner basis for $\mathbf{I}$. In particular, we have the following result.

Theorem. *Suppose $\mathbf{I}$ is a zero-dimensional ideal in $R = \mathbb{F}[x_1, \dots, x_n]$. Let $N = \dim_{\mathbb{F}}(R/\mathbf{I})$ (which is equal to the number of common solutions of $\mathbf{I}$ over the algebraic closure of $\mathbb{F}$, counting multiplicities). Then, given a Gröbner basis for $\mathbf{I}$ (under any monomial order) and a polynomial $g \in R$, Gröbner bases for $(\mathbf{I}, g)$ and $(\mathbf{I} : g)$ can be computed deterministically using $O((nN)^3)$ operations in $\mathbb{F}$.*

The time complexity claimed by the theorem is of interest only when N is small compared to n (say $N = n^{\mathcal{O}(1)}$). For when N is large or ∞ , we introduce an enhanced algorithm in Section 3. We shall define regular top-reductions and super top-reductions, as well as J-polynomials and J-signatures for any pair of polynomials. A J-polynomial means the joint of two polynomials, which is different from an S-polynomial but plays a similar role. Our algorithm is very similar to Buchberger's algorithm, where we replace S-polynomials by J-polynomials and "reduction" by "regular top-reduction". There are, however, two new features: (a) a super top-reduction is introduced to detect a useless J-polynomial, and (b) each reduction to zero gives a polynomial in $(\mathbf{I} : g)$ and is subsequently used in detecting future useless J-polynomials. We have implemented the resulting algorithm in Singular. In Section 4, we present some comparisons with F5 and F5C. Our computer experiments on several benchmark examples show that the new algorithm is more efficient, often two to ten times faster than F5 and F5C.

2. THEORY

We give a computational proof for the correspondence of linear bases for the equation (1) and the theorem mentioned in the previous section. The proof itself is more important than the theorem for our algorithm presented in the next section.

Let $\mathbf{I}$ be an arbitrary ideal in $R = \mathbb{F}[x_1, \dots, x_n]$ and g any polynomial in R . Suppose we know a Gröbner basis G for $\mathbf{I}$ with respect to some monomial order $\prec$. Then we can find the standard monomial basis for $R/\mathbf{I}$:

$$B(\mathbf{I}) = \{x^{\alpha_1} = 1, x^{\alpha_2}, \dots, x^{\alpha_N}\},$$

that is, $B(\mathbf{I})$ consists of all the monomials that are not reducible by $\text{LM}(\mathbf{I})$.¹ Then $B(\mathbf{I})$ is a linear basis for $R/\mathbf{I}$ over $\mathbb{F}$. We assume the monomials in $B(\mathbf{I})$ are ordered in increasing order, that is, $x^{\alpha_i} \prec x^{\alpha_j}$ whenever $i < j$. Please note that when $\mathbf{I}$ is not 0-dimensional, N is ∞ and it is possible that there are infinitely many monomials between some two monomials in $B(\mathbf{I})$ (especially for lex order). The following proof is for an arbitrary ideal $\mathbf{I}$.

Suppose

$$\begin{pmatrix} x^{\alpha_1} \\ x^{\alpha_2} \\ \vdots \\ x^{\alpha_N} \end{pmatrix} \cdot g \equiv \begin{pmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_N(x) \end{pmatrix} \pmod{G} \quad (3)$$

$$= A(x^{\alpha_1}, x^{\alpha_2}, \dots, x^{\alpha_N})^T, \quad (4)$$

where $h_i \in \text{span}_{\mathbb{F}}(x^{\alpha_1}, \dots, x^{\alpha_N})$, $1 \leq i \leq N$, that is, each h_i is the normal form of $x^{\alpha_i} \cdot g \pmod{G}$, and $A \in \mathbb{F}^{N \times N}$ is a matrix with the i^{th} row representing the coefficients of h_i , $1 \leq i \leq N$.

Note the matrix A in (4) has an important property that is useful for finding points (or solutions) of the algebraic variety defined by the ideal $\mathbf{I}$. In fact, when $\mathbf{I}$ is zero-dimensional, the eigenvalues of A correspond to the values of the polynomial g when evaluated at the points in the variety of $\mathbf{I}$ (and the corresponding eigenvectors are determined by the points alone, independent of g); for more details see Chapter 2 in [6].

Now apply the following row operations to both sides of (3) (equivalently (4)):

(R1) for $1 \leq i < j \leq N$ and $a \in \mathbb{F}$, subtract from the j^{th} row by the i^{th} row multiplied by a (i.e. $A_j := A_j - aA_i$),

(R2) for $a \in \mathbb{F}$ with $a \neq 0$, multiply the i^{th} row by a .

This means that we only apply row operations *downward* as one would perform Gauss elimination (to equation (4)) to get a triangular matrix. For example, suppose x^{β} is the leading monomial of $h_1(x)$. We can use $h_1(x)$ to eliminate the term x^{β} in all $h_j(x)$, $2 \leq j \leq N$. In fact, we only need to eliminate it if it's the leading term. Then continue with the leading monomial of the resulting $h_2(x)$ and so on. Since a monomial order is a well ordering, there is no infinite decreasing sequence of monomials, hence each $h_i(x)$ needs only be reduced by finitely many rows above it (even if there are infinitely many rows about the row of $h_i(x)$). Therefore, using downward row operations, the right hand side of (3) can be transformed into a quasi-triangular form, say

$$\begin{pmatrix} u_1(x) \\ u_2(x) \\ \vdots \\ u_N(x) \end{pmatrix} \cdot g \equiv \begin{pmatrix} v_1(x) \\ v_2(x) \\ \vdots \\ v_N(x) \end{pmatrix} \pmod{G}, \quad (5)$$

where $u_i(x)$ and $v_i(x)$ are in $\text{span}_{\mathbb{F}}(x^{\alpha_1}, \dots, x^{\alpha_N})$, and for each $1 \leq i, j \leq N$ with $v_i(x) \neq 0$ and $v_j(x) \neq 0$, we have $\text{LM}(v_i(x)) \neq \text{LM}(v_j(x))$, i.e. the nonzero rows of the right hand side have distinct leading monomials.

Since row operations are downward only, and the $B(\mathbf{I})$ are written in increasing order, we have that each $u_i(x)$ is monic

¹We say that a polynomial f is reducible by a set of polynomials G if $\text{LM}(f)$ is divisible by $\text{LM}(g)$ for some $g \in G$.

and

$$\text{LM}(u_i(x)) = x^{\alpha_i}, \quad 1 \leq i \leq N.$$

Let

$$\begin{aligned} G_0 &= G \cup \{u_i(x) : 1 \leq i \leq N \text{ with } v_i(x) = 0\}, \text{ and} \\ G_1 &= G \cup \{v_i(x) : 1 \leq i \leq N\}. \end{aligned}$$

Certainly, $G_1 \subseteq \langle \mathbf{I}, g \rangle$ and $G_0 \subseteq \langle \mathbf{I} : g \rangle$ (as $u_i(x) \cdot g \in \mathbf{I}$ whenever $v_i(x) = 0$). We prove the following:

- (a) G_0 is a Gröbner basis for $\langle \mathbf{I} : g \rangle$, and
- (b) G_1 is a Gröbner basis for $\langle \mathbf{I}, g \rangle$.

Since (5) is obtained from (3) by downward row operations, there is an upper triangular nonsingular matrix $M \in \mathbb{F}^{N \times N}$ (with each row containing only finitely many nonzero entries) such that

$$(u_1(x), \dots, u_N(x))^T = M(x^{\alpha_1}, \dots, x^{\alpha_N})^T,$$

and

$$(v_1(x), \dots, v_N(x))^T = M(h_1(x), \dots, h_N(x))^T.$$

Even though N could be infinite, M does have an inverse M^{-1} with each row containing only finitely many nonzero entries. For any $w(x) \in R/\mathbf{I}$, we can write it as

$$w(x) = \sum_{i=1}^N w_i x^{\alpha_i}, \quad w_i \in \mathbb{F}, \quad (6)$$

where there are only finitely many nonzero w_i 's. Let

$$(c_1, \dots, c_N) = (w_1, \dots, w_N)M^{-1} \in \mathbb{F}^N.$$

Note that the vector $(c_1, \dots, c_N)$ contains only finitely many nonzero entries, as it is a linear combination of finitely many rows of M^{-1} . Then we have

$$\begin{aligned} w(x) &= (w_1, \dots, w_N)M^{-1}M(x^{\alpha_1}, \dots, x^{\alpha_N})^T \\ &= (c_1, \dots, c_N)(u_1(x), \dots, u_N(x))^T, \end{aligned}$$

i.e.

$$w(x) = \sum_{i=1}^N c_i u_i(x), \quad (7)$$

and

$$\begin{aligned} w(x) \cdot g &= (w_1, \dots, w_N)(x^{\alpha_1}, \dots, x^{\alpha_N})^T \cdot g \\ &\equiv (w_1, \dots, w_N)M^{-1}M(h_1(x), \dots, h_N(x))^T \\ &= (c_1, \dots, c_N)(v_1(x), \dots, v_N(x))^T, \end{aligned}$$

i.e.

$$w(x) \cdot g \equiv \sum_{i=1}^N c_i v_i(x) \pmod{G}. \quad (8)$$

For (a), to prove that G_0 is a Gröbner basis for $\langle \mathbf{I} : g \rangle$, it suffices to show that each $f \in \langle \mathbf{I} : g \rangle$ can be reduced to zero by G_0 via long division. Indeed, for any $f \in \langle \mathbf{I} : g \rangle$, since G is a Gröbner basis, f can be reduced by G to some $w(x)$ as in (6). Then, by (7) and (8), we have

$$f \equiv w(x) \equiv \sum_{i=1}^N c_i u_i(x) \pmod{G},$$

and

$$f \cdot g \equiv w \cdot g \equiv \sum_{i=1}^N c_i v_i(x) \pmod{G}.$$

As $f \in \langle \mathbf{I} : g \rangle$, we have $f \cdot g \in \mathbf{I}$, so $f \cdot g \equiv 0 \pmod{G}$. This implies that $\sum_{i=1}^N c_i v_i(x) = 0$, hence $c_i = 0$ whenever $v_i(x) \neq 0$, as the nonzero $v_i(x)$'s have distinct leading monomials. Thus

$$f \equiv w(x) \equiv \sum_{u_i \in G_0} c_i u_i(x) \pmod{G}. \quad (9)$$

This implies that f can be reduced to 0 by G_0 via long division. Therefore, G_0 is a Gröbner basis for $\langle \mathbf{I} : g \rangle$.

For (b), for any $f \in \langle \mathbf{I}, g \rangle$, there exists $w(x)$ of the form (6) such that

$$f \equiv w(x) \cdot g \pmod{G}.$$

By (8),

$$f \equiv w(x) \cdot g \equiv \sum_{i=1}^N c_i v_i(x) = \sum_{v_i(x) \neq 0} c_i v_i(x) \pmod{G}. \quad (10)$$

Hence f can be reduced to 0 by $G \cup \{v_i(x) : 1 \leq i \leq N\}$ via long division. This shows that G_1 is a Gröbner basis for $\langle \mathbf{I}, g \rangle$.

Now we explicitly describe $B(\mathbf{I} : g)$ and $B(\langle \mathbf{I}, g \rangle)$, the standard monomial bases for $R/\langle \mathbf{I} : g \rangle$ and $R/\langle \mathbf{I}, g \rangle$, respectively. We first show that

$$B(\mathbf{I} : g) = \{x^{\alpha_j} : 1 \leq j \leq N \text{ and } v_j(x) \neq 0\}. \quad (11)$$

Since $\mathbf{I} \subseteq \langle \mathbf{I} : g \rangle$, we have

$$B(\mathbf{I} : g) \subseteq B(\mathbf{I}) = \{x^{\alpha_1}, \dots, x^{\alpha_N}\}.$$

Recall that $\text{LM}(u_j(x)) = x^{\alpha_j}$, $1 \leq j \leq N$. For each $1 \leq j \leq N$, if $v_j(x) = 0$, then $u_j(x) \in G_0$, so $x^{\alpha_j} \notin B(\mathbf{I} : g)$. If $v_j(x) \neq 0$, we claim that there is no $f \in \langle \mathbf{I} : g \rangle$ such that $\text{LM}(f) = x^{\alpha_j}$. Suppose otherwise. Then $f \equiv w(x) \pmod{G}$ for some $w(x)$ as in (6) and $\text{LM}(w(x)) = \text{LM}(f) = x^{\alpha_j}$. By (9), x^{α_j} must be equal to the leading monomial of some $u_i(x) \in G_0$, hence $u_j(x) \in G_0$. This contradicts the assumption that $v_j(x) \neq 0$. Hence (11) holds.

Next we claim that

$$B(\langle \mathbf{I}, g \rangle) = B(\mathbf{I}) \setminus \{\text{LM}(v_i(x)) : 1 \leq i \leq N\}. \quad (12)$$

This holds, as the equation (10) implies that the leading monomial of any $f \in \langle \mathbf{I}, g \rangle$ is either divisible by $\text{LM}(G)$ or equal to some $\text{LM}(v_i(x))$, where $v_i(x) \neq 0$, $1 \leq i \leq N$.

Now back to the proof of the theorem. The equation (2) follows from the equations (11) and (12), as the leading monomials of the nonzero $v_i(x)$ are distinct and are contained in $B(\mathbf{I})$. When $\mathbf{I}$ is zero-dimensional, the normal forms $h_i(x)$ in (3) can be computed in time cubic in nN , say by using the border basis technique [12], and Gauss elimination also needs cubic time. Hence the claimed time complexity follows.

Finally, we make a few observations concerning the above proof. They will be the basis for our algorithm below.

- $\text{LM}(u_i(x)) = x^{\alpha_i}$, so u_i is not divisible by $\text{LM}(G)$, for all $1 \leq i \leq N$. The monomial x^{α_i} is an index for the corresponding row in (3), which will be called a signature.

- For any i with $v_i(x) \neq 0$, $\text{LM}(u_i(x))$ is not divisible by $\text{LM}(G_0)$. This follows from (11).
- In the process of computing the Gröbner bases, whenever we get some $u \cdot g \equiv 0 \pmod{G}$, we add u to G_0 . So we never need to consider any u' such that $\text{LT}(u')$ is divisible by $\text{LT}(u)$.
- Both G_0 and G_1 have many redundant polynomials. We do not want to store most of them.

We need to decide which rows to store and how to perform row operations while many rows are missing. In the next section, we shall introduce regular top-reductions to emulate the row operations above and super top-reductions to detect rows that need not be stored.

3. ALGORITHM

Our algorithm computes a Gröbner basis for $(\mathbf{I} : g)$ in the process of computing a Gröbner basis for $\langle \mathbf{I}, g \rangle$. The Gröbner basis for $(\mathbf{I} : g)$ is stored in the list H in the algorithm described in figure 1. If one does not need a Gröbner basis for $(\mathbf{I} : g)$, one is free to retain only the leading monomials of H . This improves efficiency when only the Gröbner basis for $\langle \mathbf{I}, g \rangle$ is required. We provide Singular code for this version at <http://www.math.clemson.edu/~sgao/code/g2v.sing>.

Let $R = \mathbb{F}[x_1, \dots, x_n]$ with any fixed monomial order $\prec$ as above. Let $G = \{f_1, f_2, \dots, f_m\}$ be any given Gröbner basis for $\mathbf{I}$ and let $g \in R$. Consider all pairs $(u, v) \in R^2$ satisfying

$$ug \equiv v \pmod{G}. \quad (13)$$

Certainly, $G \subset \langle \mathbf{I}, g \rangle$ and $G \subset (\mathbf{I} : g)$. That is, we have the trivial solutions $(f_1, 0), (f_2, 0), \dots, (f_m, 0)$ and

$$(0, f_1), (0, f_2), \dots, (0, f_m). \quad (14)$$

The first nontrivial solution for (13) is $(1, g)$.

We need to introduce a few concepts before proceeding. For any pair $(u, v) \in R^2$, $\text{LM}(u)$ is called the **signature** of (u, v) . We make the convention that $\text{LM}(0) = 0$. Our definition of signature is similar in purpose to that of Faugère [10]. To simulate the row operation (R1), we introduce the concept of regular top-reduction. Our regular top-reduction is similar to the top-reduction used by Faugère [10], but our use of super top-reduction below seems to be new. We say that (u_1, v_1) is **top-reducible** by (u_2, v_2) if

- (i) $\text{LM}(v_2) \mid \text{LM}(v_1)$, and
- (ii) $\text{LM}(tu_2) \preceq \text{LM}(u_1)$ where $t = \frac{\text{LM}(v_1)}{\text{LM}(v_2)}$.

The corresponding **top-reduction** is then

$$(u_1, v_1) - ct(u_2, v_2) \equiv (u_1 - ct u_2, v_1 - ct v_2) \pmod{G},$$

where $c = \text{LC}(v_1)/\text{LC}(v_2)$. The effect of a top-reduction is that the leading monomial in the v -part is canceled. A top-reduction is called **super**, if

$$\text{LM}(u_1 - ct u_2) \prec \text{LM}(u_1),$$

that is, the leading monomial in the u -part is also canceled. A super top-reduction happens when

$$\text{LM}(tu_2) = \text{LM}(u_1) \text{ and } \frac{\text{LC}(u_1)}{\text{LC}(u_2)} = \frac{\text{LC}(v_1)}{\text{LC}(v_2)}.$$

A top-reduction is called **regular** if it is not super. The signature is preserved by regular top-reductions, but not by super top-reductions.

In our algorithm, we only perform regular top-reductions. We also keep all the u monic (or 0 for trivial solutions). Hence, for each regular top-reduction of (u_1, v_1) by (u_2, v_2) where u_1 and u_2 are monic, we perform the following steps:

- $u := u_1 - ct u_2$, and $v := v_1 - ct v_2$ where $t = \frac{\text{LM}(v_1)}{\text{LM}(v_2)}$ and $c = \text{LC}(v_1)/\text{LC}(v_2)$;
- if $\text{LM}(u_1) = t \text{LM}(u_2)$ then $u := u/(1 - c)$ and $v := v/(1 - c)$;
- $u := \text{Normal}(u, G)$ and $v := \text{Normal}(v, G)$, the normal forms of u and v modulo G .

Note that, if $\text{LM}(u_1) = t \text{LM}(u_2)$ and $c = 1$, then (u_1, v_1) is super top-reducible by (u_2, v_2) . We never perform super top-reductions in our algorithm. In the case that (u_1, v_1) is not regular top-reducible by other pairs known but is super top-reducible, we discard the pair (u_1, v_1) , which corresponds to a row in the equation (5) that needs not be stored (in this case v_1 is redundant in G_1).

Now we introduce a new concept of so-called J -pair for any two pairs of polynomials. Initially, we have the trivial solution pairs in (14) and the pair

$$(1, v), \quad \text{where } v = \text{Normal}(g, G), \text{ assuming } v \neq 0.$$

We find new solution pairs that are not top-reducible by the known pairs, hence must be stored. For any monomial t , consider the pair $t(1, v)$. If $t(1, v)$ is not top-reducible by any $(0, f)$ where $f \in G$, then $t(1, v) \pmod{G}$ is super top-reducible by $(1, v)$, hence we don't need to store this pair. However, if $t(1, v)$ top-reducible by some $(0, f)$ where $f \in G$, then the new pair after reduction by $(0, f)$ may not be top-reducible by $(1, v)$ any more, hence it must be stored. This means we find the smallest monomial t so that the pair $t(1, v)$ is top-reducible by some $(0, f)$. This can happen only if $t \text{LM}(v)$ is divisible by $\text{LM}(f)$ for some $f \in G$. Hence t should be such that $t \text{LM}(v) = \text{lcm}(\text{LM}(v), \text{LM}(f))$. We consider all these t given by $f \in G$. More generally, suppose we have computed a list of solution pairs

$$(u_1, v_1), (u_2, v_2), \dots, (u_k, v_k), \quad (15)$$

including the pairs in (14). We consider all pairs $t(u_i, v_i)$, $1 \leq i \leq k$, that may be top-reducible by some pair in (15). The t must come from $\text{lcm}(\text{LM}(v_i), \text{LM}(v_j))$ for some $j \neq i$. This leads us to the concept of a joint pair from any two pairs as defined below.

Let (u_1, v_1) and (u_2, v_2) be two pairs of polynomials with v_1 and v_2 both nonzero. Let

$$\text{lcm}(\text{LM}(v_1), \text{LM}(v_2)) = t, \quad t_1 = \frac{t}{\text{LM}(v_1)}, \quad t_2 = \frac{t}{\text{LM}(v_2)}.$$

Find $\max(t_1 \text{LM}(u_1), t_2 \text{LM}(u_2))$, say equal to $t_i \text{LM}(u_i)$. Then

- $t_i \text{LM}(u_i)$ is called the **J-signature** of the two pairs;
- $t_i v_i$ is called the **J-polynomial** of the two pairs;
- $t_i(u_i, v_i) = (t_i u_i, t_i v_i)$ is called the **J-pair** of the two pairs;

where J means “joint”. In comparison, the S -polynomial of v_1 and v_2 is $t_1 v_1 - (c_1/c_2)t_2 v_2$ where $c_i = \text{LC}(v_i)$. Hence our J-polynomials are related to S -polynomials. Notice that the J-signature of (u_1, v_1) and (u_2, v_2) is the same as the signature of the J-pair of (u_1, v_1) and (u_2, v_2) .

The basic idea of our algorithm is as follows. Initially, we have the pair $(1, g) \bmod G$ and the trivial pairs in (14). From these pairs, we form all J-pairs and store them in a list JP. Then take the smallest J-pair from JP and repeatedly perform regular top-reductions until it is no longer regular top-reducible. If the v part of the resulting pair is zero, then the u part is a polynomial in $(\mathbf{I} : g)$, and we store this polynomial. If the v part is nonzero, then we check if the resulting J-pair is super top-reducible. If so, then we discard this J-pair; otherwise, we add this pair to the current Gröbner basis and form new J-pairs and add them to JP. Repeat this process for each pair in JP. The algorithm is described more precisely in Figure 1 below. In the algorithm, we include two options: in first option we only keep the leading monomials of u ’s and there is no need to update u ’s in each regular top-reduction, so we compute a Gröbner basis for $\text{LM}(\mathbf{I} : g)$; in the second option, we actually update u in each regular top-reduction as specified above, so we compute a Gröbner basis for $(\mathbf{I} : g)$.

It can be proved that, when JP is empty, $\text{LM}(H)$ is a Gröbner basis for $\text{LM}(\mathbf{I} : g)$ and V is a Gröbner basis for $(\mathbf{I}, g)$, which may not be minimal. Also, for each solution (u, v) to (13), we have either $\text{LM}(u)$ is reducible by H , or (u, v) can be top-reduced to $(0, 0)$ by (U, V) (using both regular and super top-reductions). The proof of the algorithm will be included elsewhere for a more general version of this algorithm that needs not be incremental.

It should be remarked that in our algorithm we always pick the J-pair with minimal signature to reduce. This is to emulate the downward row operations of the matrix. The algorithm may not work if one uses another strategy, say picking J-pairs with minimal total degree in the v part.

4. COMPARISONS AND CONCLUSIONS

In order to determine how our algorithm compared to, say F5 and F5C, we computed Gröbner basis for various benchmark examples as provided in [8]. We used the examples and algorithm implementation for F5 and F5C provided by the URL in [8] which was all implemented in the Singular computer algebra system. Our implementation was meant to mirror the F5C implementation in terms of code structure and Singular kernel calls. For example, both implementations use the procedure “reduce” to compute normal form of a polynomial modulo a Gröbner basis. Reasonable differences were unavoidable though. For example, F5C uses Quicksort while G^2V performs one step of a Mergesort in the function “insertPairs”.

All examples considered were over the field of 7583 elements with the graded reverse lexicographic ordering. In addition to the usual wall clock times, several other measures of performance were considered, namely

1. Wall clock time (from a single run),
2. Extraneous generators,
3. Memory usage,
4. Count of J-pairs or S-pairs reduced, and

Test Case (#generators)	F5	F5C	G^2V
Katsura5 (22)	1.48	0.93	0.36
Katsura6 (41)	2.79	2.34	0.37
Katsura7 (74)	30.27	22.76	4.64
Katsura8 (143)	290.97	177.74	29.88
Schrans-Troost (128)	1180.08	299.65	21.34
F633 (76)	30.93	29.87	2.06
Cyclic6 (99)	28.44	22.06	5.65
Cyclic7 (443)	4591.20	2284.05	732.33

Test Case (#generators)	F5/ G^2V	F5C/ G^2V
Katsura5 (22)	4.11	2.58
Katsura6 (41)	7.54	6.32
Katsura7 (74)	6.52	4.91
Katsura8 (143)	9.74	5.95
Schrans-Troost (128)	55.30	14.04
F633 (76)	15.01	14.50
Cyclic6 (99)	5.03	3.90
Cyclic7 (443)	6.27	3.12

Table 1: Run-times in seconds and ratios of run-times for various test cases in Singular 3.1.0.6 on an Intel Core 2 Quad 2.66 GHz. The #generators refers to a reduced Gröbner basis.

5. Count of normal forms computed.

The run-times and ratios of run-times are presented in Table 1. One can see that, for these examples, our algorithm is two to ten times faster than F5 and F5C.

F5, F5C and our algorithm G^2V are all incremental. That is, given a list of polynomials $g_1, \dots, g_m$, a Gröbner basis is computed for $\langle g_1, g_2, \dots, g_i \rangle$ for $i = 1, 2, \dots, m$. Hence, in each iteration, all three algorithms are given a polynomial $g \in R$ and a Gröbner basis G for some ideal $\mathbf{I}$, and they compute a Gröbner basis for $(\mathbf{I}, g)$. The computed Gröbner basis is not necessarily reduced, and any redundant polynomials in the basis will result in extra S -polynomials or J-polynomials to be reduced. Fewer generators at any given time means that fewer S -polynomials or J-polynomials need to be considered. F5 uses G as it was computed, so may not be reduced, however, F5C and our algorithm always replace G by a reduced Gröbner basis. Table 2 lists the number of polynomials in the Gröbner bases that were output by each algorithm on the last iteration of each example. Computation time is not the only limiting factor in a Gröbner basis computation. Storage requirements also limit computation. Table 3 lists the maximum amount of memory each algorithm needed in the processing of examples. Again, we cannot make generalizations from the memory results because this is only one possible implementation of each algorithm in one possible CAS.

The last two criteria were also measured, but the results were not nearly as interesting. Each algorithm outperformed the other (and usually not by much) in nearly half the examples.

In conclusion, we presented a precise relationship among the degrees of the ideals $\mathbf{I}$, $(\mathbf{I}, g)$ and $(\mathbf{I} : g)$, and a connection between the Gröbner bases of $(\mathbf{I}, g)$ and $(\mathbf{I} : g)$. This allowed us to design a new algorithm, which is conceptually simpler and yet more efficient than F5 and F5C.

Input:	$G = [f_1, f_2, \dots, f_m]$, a Gröbner basis for an ideal $\mathbf{I}$, and g a polynomial.
Output:	A Gröbner basis for $\langle \mathbf{I}, g \rangle$, and a Gröbner basis for $\text{LM}(\mathbf{I} : g)$ or for $(\mathbf{I} : g)$.
Variables:	U a list of monomials for $\text{LM}(u)$ or of polynomials for u ; V a list of polynomials for v ; H a list for $\text{LM}(u)$ or u so that $u \in (\mathbf{I} : g)$ found so far, JP a list of pairs (t, i) , where t is a monomial so that $t(u_i, v_i)$ is the J-pair of (u_i, v_i) and (u_j, v_j) for some $j \neq i$. We shall refer (t, i) as the J-pair of (u_i, v_i) and (u_j, v_j) .
Step 0.	$U = [0, \dots, 0]$ with length m , and $V = [f_1, \dots, f_m]$ (so that $(u_i, v_i) = (0, f_i)$, $1 \leq i \leq m$); $H = [\text{LM}(f_1), \text{LM}(f_2), \dots, \text{LM}(f_m)]$ or $H = [f_1, f_2, \dots, f_m]$; Compute $v = \text{Normal}(g, G)$; If $v = 0$, then append 1 to H and return V and H (stop the algorithm); else append 1 to U and v to V ; $JP = []$, an empty list; For each $1 \leq i \leq m$, compute the J-pair of the two pairs $(u_{m+1}, v_{m+1}) = (1, v)$ and $(u_i, v_i) = (0, f_i)$, such a J-pair must be of the form $(t_i, m+1)$, insert $(t_i, m+1)$ into JP whenever t_i is not reducible by H . (store only one J-pair for each distinct J-signature).
Step 1.	Take a minimal (in signature) pair (t, i) from JP , and delete it from JP .
Step 2.	Reduce the pair $t(u_i, v_i)$ repeatedly by the pairs in (U, V) , using regular top-reductions, say to get (u, v) , which is not regular top-reducible.
Step 3a.	If $v = 0$, then append $\text{LM}(u)$ or u to H and delete every J-pair (t, ℓ) in JP whose signature $t\text{LM}(u_\ell)$ is divisible by $\text{LM}(u)$.
Step 3b.	If $v \neq 0$ and (u, v) is super top-reducible by some pair (u_j, v_j) in (U, V) , then discard the pair (t, i) .
Step 3c.	Otherwise, append u to U and v to V , form new J-pairs of (u, v) and (u_j, v_j) , $1 \leq j \leq \#U - 1$, and insert into JP all such J-pairs whose signature are not reducible by H (store only one J-pair for each distinct J-signature).
Step 4.	While JP is not empty, go to step 1.
Return:	V and H .

Figure 1: Algorithm

Test Case (#generators)	F5	F5C	G ² V
Katsura5 (22)	61	44	63
Katsura6 (41)	74	65	52
Katsura7 (74)	185	163	170
Katsura8 (143)	423	367	335
Schrans-Troost (128)	643	399	189
F633 (76)	237	217	115
Cyclic6 (99)	202	183	146
Cyclic7 (443)	1227	1006	658

Table 2: The number of generators in the Gröbner basis in the last iteration but before computing a reduced Gröbner basis. Of course, F5 never computes the reduced Gröbner basis.

Test Case (#generators)	F5	F5C	G ² V
Katsura5 (22)	1359	828	1255
Katsura6 (41)	1955	1409	1254
Katsura7 (74)	8280	4600	5369
Katsura8 (143)	40578	20232	20252
Schrans-Troost (128)	130318	50566	32517
F633 (76)	3144	2720	2824
Cyclic6 (99)	2749	2280	1789
Cyclic7 (443)	48208	23292	24596

Table 3: The maximum amount of memory (in KiB) Singular 3.1.0.6 used from startup to the conclusion of the Gröbner basis computation. Memory amounts obtained with “memory(2);”.

5. REFERENCES

- [1] B. BUCHBERGER, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, PhD thesis, Innsbruck, 1965.
- [2] B. BUCHBERGER, “A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Basis,” In Proc. EUROSAM 79 (1979), vol. 72 of Lect. Notes in Comp. Sci., Springer Verlag, 3–21.
- [3] B. BUCHBERGER, “Gröbner Bases : an Algorithmic Method in Polynomial Ideal Theory,” In Recent trends in multidimensional system theory, Ed. Bose, 1985.
- [4] J. BUCHMANN, D. CABARCAS, J. DING, M. S. E. MOHAMED, “Flexible Partial Enlargement to Accelerate Grobner Basis Computation over F2,” AFRICACRYPT 2010, May 03-06, 2010, Stellenbosch, South Africa, to be published in LNCS by Springer
- [5] N. COURTOIS, A. KLIMOV, J. PATARIN, AND A. SHAMIR, “Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations,” in Proceedings of International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT), volume 1807 of Lecture Notes in Computer Science, pages 392–407, Bruges, Belgium, May 2000. Springer.
- [6] D. COX, J. LITTLE AND D. O’SHEA, *Using algebraic geometry*, Graduate Texts in Mathematics, 185. Springer-Verlag, New York, 1998.
- [7] J. DING, J. BUCHMANN, M. S. E. MOHAMED, W. S. A. M. MOHAMED, R. WEINMANN, “Mutant XL,” First International Conference on Symbolic Computation and Cryptography, SCC 2008
- [8] C. EDER AND J. PERRY, “F5C: a variant of Faugère’s F5 algorithm with reduced Gröbner bases,” arXiv: 0906.2967v5, July 2009.
- [9] J.-C. FAUGÈRE, “A new efficient algorithm for computing Gröbner bases (F4),” Effective methods in algebraic geometry (Saint-Malo, 1998), J. Pure Appl. Algebra 139 (1999), no. 1-3, 61–88.
- [10] J.-C. FAUGÈRE, “A new efficient algorithm for computing Gröbner bases without reduction to zero (F5),” Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, 75–83 (electronic), ACM, New York, 2002.
- [11] A. JOUX AND V. VITSE, “A variant of the F4 algorithm,” preprint 2010.
<http://eprint.iacr.org/2010/158>.
- [12] A. KEHREIN AND M. KREUZER, “Computation of Border Bases,” J. Pure Appl. Algebra 205 (2006), 279–295.
- [13] D. LAZARD, “Gaussian Elimination and Resolution of Systems of Algebraic Equations,” in Proc. EUROCAL 83 (1983), vol. 162 of Lect. Notes in Comp. Sci, 146–157.
- [14] F. MACAULAY, “Some formulae in elimination,” Proceedings of London Mathematical Society (1902), 3–38.
- [15] M. S. E. MOHAMED, D. D. CABARCAS, J. DING, J. BUCHMANN AND S. BULYGIN, “MXL3: An efficient algorithm for computing Groebner bases of zero-dimensional ideals,” the 12th International Conference on Information Security and Cryptology (ICISC 2009), Dec. 2009, Seoul, Korea. (To be published in LNCS by Springer).
- [16] M. S. E. MOHAMED, W. S. A. E. MOHAMED, J. DING, J. BUCHMANN, “MXL2: Solving Polynomial Equations over GF(2) Using an Improved Mutant Strategy,” PQCrypto 2008: 203-215, LNCS 5299, Springer 2008
- [17] Y. SUN AND D.K. WANG, “A New Proof of the F5 Algorithm,” preprint 2009.
<http://www.mmrc.iss.ac.cn/pub/mm28.pdf/06-F5Proof.pdf>

Degree Bounds for Gröbner Bases of Low-Dimensional Polynomial Ideals

Ernst W. Mayr
Technische Universität München
Boltzmannstr. 3
D-85748 Garching
mayr@in.tum.de

Stephan Ritscher
Technische Universität München
Boltzmannstr. 3
D-85748 Garching
ritsches@in.tum.de

ABSTRACT

Let $\mathbb{K}[X]$ be a ring of multivariate polynomials with coefficients in a field $\mathbb{K}$, and let $f_1, \dots, f_s$ be polynomials with maximal total degree d which generate an ideal I of dimension r . Then, for every admissible ordering, the total degree of polynomials in a Gröbner basis for I is bounded by $2 \left(\frac{1}{2} d^{n-r} + d \right)^{2^r}$. This is proved using the cone decompositions introduced by Dubé in [5]. Also, a lower bound of similar form is given.

Categories and Subject Descriptors

I.1.1 [Symbolic and Algebraic Manipulation]: Expressions and Their Representation—*Representations (general and polynomial)*; G.2.1 [Discrete Mathematics]: Combinatorics—*Counting problems*

General Terms

Theory

Keywords

multivariate polynomial, Gröbner basis, polynomial ideal, ideal dimension, complexity

1. INTRODUCTION

Gröbner bases are a very powerful tool in computer algebra which was introduced by Buchberger [2]. Many problems that can be formulated in the language of polynomials can be easily solved once a Gröbner basis has been computed. This is because Gröbner bases allow quick ideal consistency checks, ideal membership tests, and ideal equality tests, among others.

Unfortunately, the computation of a Gröbner basis can be very expensive. The problem is exponential space complete, which was shown in [13] and [11]. Interestingly, both the upper and the lower bound are obtained by considering polynomials of high degree in the ideal. So knowing good

bounds for the degrees of polynomials in the Gröbner basis means also knowing the complexity of its calculation.

In [13] and [14] it was shown that, in the worst case, the degree of polynomials in a Gröbner basis is at least doubly exponential in the number of indeterminates of the polynomial ring. [1], [7], and [14] provide a doubly exponential upper degree bound as explained in the introduction of [5]. [5] gives a combinatorial proof of an improved upper bound.

For zero-dimensional ideals, the bounds are smaller by a magnitude. The well-known theorem of Bézout (cf. [16]) immediately implies a singly exponential upper degree bound. For graded monomial orderings the degrees are even bounded polynomially, as proved in [12]. Both bounds are exact, and the examples providing lower bounds are folklore (cf. [14]).

This suggests that also ideals with small non-zero dimension permit better degree bounds than in the general case. Furthermore, in [9] an ideal membership test was provided with space complexity exponential only in the dimension of the ideal. This result anticipated a degree bound of the form shown in this paper.

The remainder of the paper is organized as follows. In the second section notation for polynomial ideals and Gröbner bases will be fixed. We do not give proofs or comprehensive explanations. For a detailed introduction and accompanying proofs we refer to the literature. The third chapter contains the main result of this paper, the upper degree bound depending on the ideal dimension. Since the proof uses cone decompositions as defined by Dubé in [5], we first review these techniques. Then we explain how to adapt this approach to get a dependency on the ideal dimension and derive the upper degree bound. Finally we demonstrate how to use the results from [13] and [14] to obtain a lower bound of similar form.

Credits

We wish to thank the referees for their detailed feedback. Particular thanks are due to one of them for showing how to tighten our bound.

2. NOTATION

In this chapter, we define the notation that will be used throughout the paper. For a more detailed introduction to polynomial algebra, the reader may consult [3] and [4].

2.1 Polynomial Ideals

Consider the ring $\mathbb{K}[X]$ of polynomials in the variables $X = \{x_1, \dots, x_n\}$. The (total) degree of a monomial is $\deg(x_1^{e_1} \cdots x_n^{e_n}) = e_1 + \dots + e_n$. A polynomial is called

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

homogeneous if all its monomials have the same degree. Every polynomial $f \neq 0$ permits a unique representation $f = f_0 + \dots + f_d$, $f_d \neq 0$, with f_k being homogeneous of degree k , the so-called *homogeneous components* of f . The *homogenization* of f with respect to a new variable x_{n+1} is defined by ${}^h f = x_{n+1}^d f_0 + x_{n+1}^{d-1} f_1 + \dots + f_d$. A set $S \subset \mathbb{K}[X]$ is called *homogeneous* if for every polynomial $f \in S$ also its homogeneous components f_k are elements of S .

Throughout the paper we assume some arbitrary but fixed *admissible* monomial ordering (cf. [5], §2.1). Therefore we won't keep track of it in the notation. The largest monomial occurring in a polynomial f is called *leading monomial* and denoted by $\text{LM}(f)$.

$\langle f_1, \dots, f_s \rangle$ denotes the *ideal* $\{\sum_{i=1}^s a_i f_i : a_i \in \mathbb{K}[X]\}$ generated by $F = \{f_1, \dots, f_s\}$. G is a *Gröbner basis* of the ideal I if $\langle G \rangle = I$ and $\langle \text{LM}(G) \rangle = \langle \text{LM}(I) \rangle$.

$\text{nf}_I(f)$ denotes the *normal form* of f , which, for a fixed monomial ordering, is the unique irreducible polynomial fulfilling $\text{nf}_I(f) \equiv f \pmod{I}$. The set of all normal forms is denoted by N_I . Since the normal forms are unique, we have the direct sum $\mathbb{K}[X] = I \oplus N_I$. For details see [5], §2.1.

Unless stated differently, we will consider an ideal I generated by homogeneous polynomials $f_1, \dots, f_s$ with degrees $d_1 \geq \dots \geq d_s$.

2.2 Hilbert Functions

Let $T \subset \mathbb{K}[X]$ be homogeneous and $T_z = \{f \in T : f \text{ homogeneous with } \deg(f) = z \text{ or } f = 0\}$ the homogeneous polynomials of T of degree z . Then the *Hilbert function* of T is defined as

$$\varphi_T(z) = \dim_{\mathbb{K}}(T_z),$$

i.e. the vector space dimension of T_z over the field $\mathbb{K}$. It easily follows from the dimension theorem for direct sums that

$$\varphi_{S \oplus T}(z) = \varphi_S(z) + \varphi_T(z).$$

It is well-known that, for large values of z , the Hilbert functions $\varphi_I(z)$ and $\varphi_{N_I}(z)$ of a homogeneous ideal I and its normal forms N_I are polynomials. These polynomials, known as *Hilbert polynomials*, will be denoted by $\bar{\varphi}_I(z)$ and $\bar{\varphi}_{N_I}(z)$, respectively.

2.3 Ideal Dimension

The *dimension* of an homogeneous ideal can be defined in many equivalent ways (cf. [3], §9). The following definition turns out to be the most suitable for our purpose.

$$\dim(I) = \deg(\bar{\varphi}_{N_I}) + 1$$

with $\deg(0) = -1$. We add 1 to the degree in order to obtain the affine instead of the projective dimension. This simplifies the presentation which is inherently affine.

Since we will have to deal with ideal dimensions and vector space dimensions, we will write $\dim(I)$ for the former and $\dim_{\mathbb{K}}(I)$ for the latter in order to avoid confusion.

2.4 Regular Sequences

A sequence $(g_1, \dots, g_t)$ with $g_k \in \mathbb{K}[X]$ is called *regular sequence* (cf. [6]) if

- g_k is a nonzerodivisor in $\mathbb{K}[X]/\langle g_1, \dots, g_{k-1} \rangle$, for all $1 \leq k \leq t$, and
- $\mathbb{K}[X] \neq \langle g_1, \dots, g_t \rangle$.

A nice, well-known property of regular sequences is that their Hilbert polynomial only depends on the degrees of the polynomials and the number of indeterminates.

PROPOSITION 2.1. *Let $(g_1, \dots, g_t)$ with $g_k \in \mathbb{K}[X]$ be a homogeneous regular sequence with degrees $d_1 \geq \dots \geq d_t$ and $J = \langle g_1, \dots, g_t \rangle$. Then $N_J \cong \mathbb{K}[X]/J$ have (for any term ordering) a Hilbert function which only depends on n , t , and $d_1, \dots, d_t$. The Hilbert function and the Hilbert polynomial are equal for $z > d_1 + \dots + d_t - n$.*

PROOF. See [10], §5.2B and §5.4B. $\square$

We are given now an ideal I of dimension r and want to embed a regular sequence which is as long as possible. It turns out that the length of this sequence is always $n - r$.

PROPOSITION 2.2 (CF. SCHMID 1995). *Let $\mathbb{K}$ be an infinite field and $I \subsetneq \mathbb{K}[X]$ an ideal generated by homogeneous polynomials $f_1, \dots, f_s$ with degrees $d_1 \geq \dots \geq d_s$ and $\dim(I) \leq r$. Then there are a permutation σ of $\{1, \dots, s\}$ and homogeneous $a_{k,i} \in \mathbb{K}[X]$ such that*

$$g_k = \sum_{i=\sigma(k)}^s a_{k,i} f_i$$

for $k = 1, \dots, n - r$ form a regular sequence of homogeneous polynomials, and $\deg(g_k) = d_{\sigma(k)}$.

PROOF. See [15], Lemma 2.2. It's an extension to the homogeneous case. Since in the ring $\mathbb{K}[X]$ any permutation of a regular sequence is regular, one can choose $\sigma = \text{id}$. $\square$

3. UPPER DEGREE BOUND

3.1 Cone Decompositions

The upper degree bound presented in this paper is based on the concept of cone decompositions introduced in [5]. This section will summarize the results that will be used leaving out the proofs which can be found in the original paper [5].

For a homogeneous polynomial h and a set of variables $U \subset X$, the corresponding *cone* is denoted by $C = C(h, U) = h\mathbb{K}[U]$. For succinctness, by the *degree* of a cone C we mean the degree of its apex, i.e., $\deg(C) = \deg(h)$. Similarly, we call the cardinality of U the *dimension* of the cone, i.e. $\dim(C) = \#U$. Note that h and U are uniquely determined by C as a set. Since we will not describe algorithms as in [5], we don't need to talk about pairs of h and U as a representation of the cone.

One of the most important reasons for working with cones is that their Hilbert functions can be easily calculated. For a cone C of dimension 0, we have

$$\varphi_C(z) = \begin{cases} 0 & \text{for } z \neq \deg(C) \\ 1 & \text{for } z = \deg(C) \end{cases},$$

for cones of dimension greater than zero

$$\varphi_C(z) = \begin{cases} 0 & \text{for } z < \deg(C) \\ \binom{z - \deg(C) + \dim(C) - 1}{\dim(C) - 1} & \text{for } z \geq \deg(C) \end{cases}.$$

Since we can handle the Hilbert functions of direct sums, we want to express the spaces we deal with as direct sums of cones.

DEFINITION 3.1 (DUBÉ 1990). Let T be a vector space and $T = \bigoplus_{i=1}^l C_i$ a direct decomposition into cones C_i . Then we call $P = \{C_i : i = 1, \dots, l\}$ a cone decomposition of T . We will use the notation $\deg(P) = \max\{\deg(C) : C \in P\}$.

Obviously

$$\bar{\varphi}_T(z) = \sum_{C \in P} \bar{\varphi}_C(z).$$

In a slight abuse of notation we also write $\varphi_P(z)$ for $\varphi_T(z)$ (respectively $\bar{\varphi}_P(z)$ for $\bar{\varphi}_T(z)$) if P is a cone decomposition of T . Our final interest will not be the Hilbert function of a cone decomposition but its Hilbert polynomial. Therefore we define $P^+ = \{C \in P : \dim(C) > 0\}$, the subset of cones with dimension greater 0. One can easily check that the polynomial part of zero-dimensional cones is 0. Therefore

$$\bar{\varphi}_P(z) = \bar{\varphi}_{P^+}(z) = \sum_{C \in P^+} \bar{\varphi}_C(z).$$

Here

$$\begin{aligned} \bar{\varphi}_C(z) &= \binom{z - \deg(C) + \dim(C) - 1}{\dim(C) - 1} \\ &= \frac{(z - \deg(C) + \dim(C) - 1) \cdots (z - \deg(C) + 1)}{(\dim(C) - 1) \cdots 1}. \end{aligned}$$

We want to consider cone decompositions whose Hilbert polynomial has a nice representation which is interlinked with the maximal degree of a reduced Gröbner basis. The first step towards this is the following definition.

DEFINITION 3.2 (DUBÉ 1990). A cone decomposition P is k -standard for some $k \in \mathbb{N}$ if

- $C \in P^+$ implies $\deg(C) \geq k$ and
- for all $C \in P^+$ and for all $k \leq d \leq \deg(C)$, there exists a cone $C^{(d)} \in P$ with degree d and dimension at least $\dim(C)$.

Note that P is k -standard for all k if and only if $P^+ = \emptyset$. Otherwise it can be k -standard for at most one k , namely the minimal degree of the cones in P^+ . Furthermore, the union of k -standard decompositions is k -standard, again.

LEMMA 3.3 (DUBÉ 1990). Every k -standard cone decomposition P may be refined into a $(k+1)$ -standard cone decomposition P' with $\deg(P) \leq \deg(P')$ and $\deg(P^+) \leq \deg(P'^+)$.

PROOF. See [5], Lemma 3.1. $\square$

Dubé was able to construct such cone decompositions for the set of normal forms of an ideal.

PROPOSITION 3.4 (DUBÉ 1990). For any homogeneous ideal $I \subset \mathbb{K}[X]$ and any monomial ordering $\prec$, there is a 0-standard cone decomposition Q of N_I such that $\deg(Q) + 1$ is an upper bound on the degrees of polynomials required in a Gröbner basis of I .

PROOF. See [5], Theorem 4.11. $\square$

The next step in [5] is a worst case construction. The question that arises is: How large can the degrees of the cones in Q and thus the degrees in the Gröbner basis be? We know that a k -standard cone decomposition P contains at least one cone in each degree between k and the maximal degree. So in the worst case there would be exactly one cone in each degree.

DEFINITION 3.5 (DUBÉ 1990). A k -standard cone decomposition P is k -exact if $\deg(C) \neq \deg(C')$ for all $C \neq C' \in P^+$.

Since k -exact cone decompositions are also k -standard, the cones of higher degrees have lower dimensions, i.e., $C, C' \in P, \deg(C) > \deg(C')$ implies $\dim(C) \leq \dim(C')$.

Since one can split a cone into a cone of dimension 0 and same degree and cones of higher degrees, one can refine a k -standard cone decomposition such that it becomes k -exact. Dubé gives an algorithmic proof herefore.

LEMMA 3.6 (DUBÉ 1990). Every k -standard cone decomposition P may be refined into a k -exact cone decomposition P' with $\deg(P) \leq \deg(P')$ and $\deg(P^+) \leq \deg(P'^+)$.

PROOF. See [5], Lemma 6.3. $\square$

A nice side effect of this worst case construction is that we can easily calculate the Hilbert polynomial of an exact cone decomposition P of some space T . Herefore we need the following notion.

DEFINITION 3.7 (DUBÉ 1990). Let P be a k -exact cone decomposition. If $P^+ = \emptyset$, let $k = 0$. Then the Macaulay constants of P are defined as

$$a_i = \max\{k, \deg(C) + 1 : C \in P^+, \dim(C) \geq i\}$$

for $i = 0, \dots, n + 1$.

Note that the definition looks slightly different from the one given in [5], but is equivalent to it. This definition implies $\max\{k, \deg(P^+)\} = a_0 \geq \dots \geq a_n \geq a_{n+1} = k$. Now

$$\bar{\varphi}_T(z) = \sum_{i=1}^n \sum_{c=a_{i+1}}^{a_i-1} \binom{z - c + i - 1}{i - 1}.$$

Some lengthy calculations in [5] finally yield

LEMMA 3.8 (DUBÉ 1990). Given a k -exact decomposition P of some space T , the Hilbert polynomial of T is given by

$$\bar{\varphi}_T(z) = \binom{z - k + n}{n} - 1 - \sum_{i=1}^n \binom{z - a_i + i - 1}{i}. \quad (1)$$

The Macaulay constants (except a_0) may be deduced from Hilbert polynomial and thus depend only on $\bar{\varphi}_T$ and not on the chosen decomposition.

PROOF. See [5], Lemma 7.1. $\square$

We are going to apply this result to an ideal generated by an exact sequence.

COROLLARY 3.9. If P is a k -exact decomposition of N_J for an ideal J generated by a homogeneous regular sequence $g_1, \dots, g_t$ of degrees $d_1, \dots, d_t$. Then the Macaulay constants (except a_0) depend only on n, t , and $d_1, \dots, d_t$, and neither on the chosen monomial ordering nor on the generators of J .

PROOF. This is a direct consequence of Proposition 2.1 and Lemma 3.8. $\square$

3.2 A New Decomposition

In order to bound the Macaulay constants of a homogeneous ideal $I = \langle f_1, \dots, f_s \rangle$, Dubé uses the direct decompositions

$$\mathbb{K}[X] = I \oplus N_I$$

and

$$I = \langle f_1 \rangle \oplus \bigoplus_{i=2}^s f_i \cdot N_{\langle f_1, \dots, f_{i-1} \rangle : f_i},$$

where $H : g = \{f : fg \in H\}$ is a special case of the ideal quotient. The Hilbert functions of $\mathbb{K}[X]$ and $\langle f_1 \rangle$ are easily determined, and for all other summands one can calculate exact cone decompositions using the theory explained in the previous section. In Dubé's construction, the Macaulay constants achieve their worst case bound in the zero-dimensional case. Therefore we are going to use a slightly different decomposition.

So let I be an r -dimensional ideal generated by homogeneous polynomials $f_1, \dots, f_s$ with degrees $d_1 \geq \dots \geq d_s$. According to Proposition 2.2, there is a regular sequence $g_1, \dots, g_{n-r} \in I$ with $\deg(g_k) = d_k$. First we prove a decomposition along the lines of Dubé, but starting from $J = \langle g_1, \dots, g_{n-r} \rangle$ instead of $\langle f_1 \rangle$.

LEMMA 3.10. *With the stated hypotheses,*

$$I = J \oplus \bigoplus_{i=1}^s \text{nf}_J(f_i) \cdot N_{J_i : \text{nf}_J(f_i)} \quad (2)$$

with $J_k = \langle g_1, \dots, g_{n-r}, f_1, \dots, f_{k-1} \rangle$.

PROOF. To prove this, we inductively show

$$J_{k+1} = J \oplus \bigoplus_{i=1}^k \text{nf}_J(f_i) \cdot N_{J_i : \text{nf}_J(f_i)}$$

for $k = 0, \dots, s-1$. The equality $I = J_s$ then yields the stated result.

The " $\supset$ "-inclusion is clear since $f_j, g_j \in I$. For the other inclusion, the case $k = 0$ is trivial. So assume $k > 0$. Let $f \in J_{k+1}$ and thus $f = f' + a \cdot f_k = (f' + a \cdot (f_k - \text{nf}_J(f_k))) + a \cdot \text{nf}_J(f_k)$ with $f', a \cdot (f_k - \text{nf}_J(f_k)) \in J_k$. We rewrite

$$a = (a - \text{nf}_{J_k : \text{nf}_J(f_k)}(a)) + \text{nf}_{J_k : \text{nf}_J(f_k)}(a),$$

which yields

$$a \cdot \text{nf}_J(f_k) \in (J_k : \text{nf}_J(f_k)) \cdot \text{nf}_J(f_k) + N_{J_k : \text{nf}_J(f_k)} \cdot \text{nf}_J(f_k).$$

Since $(J_k : \text{nf}_J(f_k)) \cdot \text{nf}_J(f_k) \subset J_k$, we get $f \in J_k + \text{nf}_J(f_k) \cdot N_{J_k : \text{nf}_J(f_k)}$ and inductively J_{k+1} of the stated form. It remains to show that the sum is direct. But this is clear since

$$J_k \cap \text{nf}_J(f_k) \cdot N_{J_k : \text{nf}_J(f_k)} \subset J_k \cap N_{J_k} = \{0\}.$$

□

Now we are going to construct cone decompositions for the parts of (2).

PROPOSITION 3.11. *With the stated hypotheses, any 0-standard decomposition Q of N_I may be completed into a d_1 -standard decomposition P of N_J such that $\deg(Q) \leq \deg(P)$.*

PROOF. By Proposition 3.4, we can construct 0-standard cone decompositions Q_k of $N_{J_k : \text{nf}_J(f_k)}$. Then $f_k \cdot Q_k$ are d_k -standard cone decompositions of $f_k \cdot N_{J_k : \text{nf}_J(f_k)}$. By Lemma

3.3, $Q, Q_1, \dots, Q_s$ can be refined into d_1 -standard cone decompositions $Q', Q'_1, \dots, Q'_s$. Since

$$\mathbb{K}[X] = J \oplus \bigoplus_{i=1}^s \text{nf}_J(f_i) \cdot N_{J_i : \text{nf}_J(f_i)} \oplus N_I,$$

the union

$$P' = Q' \cup Q'_1 \cup \dots \cup Q'_s$$

is a d_1 -standard cone decomposition of N_J . By Lemma 3.6, this can be refined to a d_1 -exact cone decomposition P of N_J with maximal degree $\deg(Q) \leq \deg(P)$. Thus the maximal degree of cones in P is also an upper bound on the Gröbner basis degree. □

All Macaulay constants of a cone decomposition P of N_J except $a_0 = \deg(P)$ are determined by the Hilbert polynomial. But, because of Proposition 3.4 and 3.11, $\deg(P)$ is what we are actually interested in. Thus we want to bound a_0 using the other Macaulay constants which can be determined from the Hilbert polynomial (Corollary 3.9).

LEMMA 3.12. *Let J be the ideal generated by the regular sequence $g_1, \dots, g_{n-r}$ with degrees $d_1, \dots, d_{n-r}$, P be a cone decomposition of N_J and $a_0, \dots, a_{n+1}$ the corresponding Macaulay constants. Then*

$$a_0 \leq \max\{a_1, d_1 + \dots + d_{n-r} - n\}.$$

PROOF. Consider

$$\mathbb{K}[X] = J \oplus \bigoplus_{C \in P} C.$$

We know that the Hilbert functions (Hilbert polynomials) of the left hand and the right hand side agree. Furthermore, for z large enough (by Proposition 2.1, $z > d_1 + \dots + d_{n-r} - n$ suffices) $\varphi_{\mathbb{K}[X]}(z) = \bar{\varphi}_{\mathbb{K}[X]}(z)$ and $\varphi_J(z) = \bar{\varphi}_J(z)$. This yields for $a_1 \leq z < a_0$:

$$\begin{aligned} \#\{C \in P : \dim(C) = 0, \deg(C) = z\} &= \varphi_P(z) - \bar{\varphi}_P(z) \\ &= (\varphi_{\mathbb{K}[X]}(z) - \varphi_J(z)) - (\bar{\varphi}_{\mathbb{K}[X]}(z) - \bar{\varphi}_J(z)) = 0 \end{aligned}$$

Thus there are no cones with degree greater or equal $\max\{a_1, d_1 + \dots + d_{n-r} - n\}$ which implies the statement. □

As a consequence of Proposition 3.11, Corollary 3.9 and Lemma 3.12, we can choose a nice ideal J for the further considerations - independent of I .

COROLLARY 3.13. *Let Q be a 0-standard cone decomposition of N_I for some ideal I and some fixed admissible monomial ordering. If I has dimension r and is generated by homogeneous polynomials $f_1, \dots, f_s$ of degrees $d_1 \geq \dots \geq d_s$, then*

$$\deg(Q) \leq \max\{\deg(P^+), d_1 + \dots + d_{n-r} - n\}$$

where P is a d_1 -exact cone decomposition of N_J and J is the ideal $\langle x_{r+1}^{d_1}, \dots, x_n^{d_{n-r}} \rangle$.

PROOF. By Proposition 3.11, we can extend a 0-standard cone decomposition Q of N_I to an d_1 -exact cone decomposition P' of $N_{J'}$ with $\deg(Q) \leq \deg(P')$ for $J' \subset I$ being generated by a homogeneous regular sequence of length $n-r$ and degrees $d_1, \dots, d_{n-r}$. By Lemma 3.12, $\deg(P') = a_0$ can be bounded by $\deg(P'^+) = a_1$. By Corollary 3.9, the Macaulay constants a_k of P' (except a_0) only depend on

$n, n-r$, and $d_1, \dots, d_{n-r}$. The ideal $J = \langle x_{r+1}^{d_1}, \dots, x_n^{d_{n-r}} \rangle$ is obviously a r -dimensional ideal generated by a homogeneous regular sequence with the same degrees. Thus a d_1 -exact cone decomposition of N_J (which exists by Proposition 3.4) has the same Macaulay constants (except a_0) and thus $\deg(P'^+) = \deg(P^+)$. $\square$

EXAMPLE 3.14. *Before we continue the proof and bound the Macaulay constants in the next section, we want to illustrate that the Macaulay constants are independent of the ideals I and J . We will work in the ring $\mathbb{K}[x_1, x_2, x_3]$ for this example, i.e., $n = 3$. First we consider the very simple ideal $I = \langle x_1^2 \rangle$ (i.e., $d_1 = 2$), which has dimension $r = 2$, and the regular sequence $g_1 = x_1^2$. Using the concepts of this section and the algorithms from [5], implemented in Singular [8], we obtain an exact cone decomposition P of $N_{(g_1)}$. Due to its size we only list the cones of positive dimension:*

$$P^+ = \{C(x_2^2, \{x_2, x_3\}), C(x_1x_2^2, \{x_2, x_3\}), C(x_2x_3^3, \{x_3\}), C(x_3^5, \{x_3\}), C(x_1x_2x_3^4, \{x_3\}), C(x_1x_3^6, \{x_3\})\}$$

Now we do the same for $I' = \langle x_1^2 - x_1x_2, x_1x_2 + x_1x_3 \rangle$ and the regular sequence $g'_1 = x_1^2 - x_2x_3$.

$$P'^+ = \{C(x_2^2, \{x_2, x_3\}), C(x_1x_2^2 + x_1x_2x_3, \{x_2, x_3\}), C(x_2x_3^3, \{x_3\}), C(x_3^5, \{x_3\}), C(x_1x_3^5, \{x_3\}), C(x_1x_2x_3^5 + x_1x_3^6, \{x_3\})\}$$

Both P and P' are exact cone decompositions with the same parameters n, r, d_1 and thus - as expected - have the same Macaulay constants:

$$a_0 = 8, a_1 = 8, a_2 = 4, a_3 = 2.$$

3.3 Macaulay Constants

By Corollary 3.13, it suffices to bound the Macaulay constant a_1 of a d_1 -exact cone decomposition of N_J for the ideal $J = \langle x_{r+1}^{d_1}, \dots, x_n^{d_{n-r}} \rangle$, which will be fixed for the remainder of this section.

The special shape of this ideal allows to dramatically simplify the corresponding proof in Dubé's paper which does not make any assumption on the ideal, except that it is generated by monomials. Nevertheless the bound we will obtain applies to any ideal by preceding corollary.

From $r = \dim(J) = \deg(\bar{\varphi}_J) + 1$, one immediately deduces:

$$\text{LEMMA 3.15. } a_n = \dots = a_{r+1} = d_1.$$

In order to determine the remaining Macaulay constants, we have to determine N_J . For the ideal J we chose, this is

$$N_J = T_r \otimes \mathbb{K}[x_1, \dots, x_r],$$

where the vector space T_r is given by

$$T_r = \text{span}_{\mathbb{K}} \{m \in \mathbb{K}[x_{r+1}, \dots, x_n] : m \text{ monomial, } x_i^{d_i-r} \nmid m \text{ for } i = r+1, \dots, n\} \quad (3)$$

and $A \otimes B$ denotes the tensor product of A and B , i.e., the vector space generated by $\{ab : a \in A, b \in B\}$. we need the following observation:

LEMMA 3.16. *Any cone decomposition P_k of a vector space $T_k \otimes \mathbb{K}[x_1, \dots, x_k]$, T_k generated by monomials, has exactly $\dim_{\mathbb{K}}(T_k)$ cones of dimension k .*

PROOF. The key is to look at the Hilbert polynomials. We easily see that, for a monomial basis $\{t_1, \dots, t_l\}$ of T_k ,

$$T_k \otimes \mathbb{K}[x_1, \dots, x_k] = t_1 \mathbb{K}[x_1, \dots, x_k] \oplus \dots \oplus t_l \mathbb{K}[x_1, \dots, x_k]$$

and

$$\bar{\varphi}_{T_k \otimes \mathbb{K}[x_1, \dots, x_k]}(z) = \sum_{i=1}^l \binom{z - \deg(t_i) + k - 1}{k - 1}.$$

On the other hand, the Hilbert polynomial of the cone decomposition P_k is

$$\bar{\varphi}_{P_k}(z) = \sum_{C \in P_k} \binom{z - \deg(C) + \dim(C) - 1}{\dim(C) - 1}.$$

Since P_k is a cone decomposition of $T_k \otimes \mathbb{K}[x_1, \dots, x_k]$, we have $\bar{\varphi}_{T_k \otimes \mathbb{K}[x_1, \dots, x_k]}(z) = \bar{\varphi}_{P_k}(z)$. Now compare the coefficients of z^{k-1} of both polynomials. Since P_k only contains cones of dimension at most k , this yields

$$\sum_{i=1}^l \frac{1}{(k-1)!} = \sum_{\substack{C \in P \\ \dim(C)=k}} \frac{1}{(k-1)!}$$

and thus $\#\{C \in P : \dim(C) = k\} = l = \dim_{\mathbb{K}}(T_k)$. $\square$

Looking at the explicit formula (3) for T_r , one obtains $\dim(T_r) = d_1 \cdots d_{n-r}$ and thus:

$$\text{LEMMA 3.17. } a_r = d_1 \cdots d_{n-r} + d_1.$$

Now we construct a d_1 -exact cone decomposition with a special form. This allow us to bound the further Macaulay constants.

LEMMA 3.18. *There exist a d_1 -exact cone decomposition P of N_J ($J = \langle x_{r+1}^{d_1}, \dots, x_n^{d_{n-r}} \rangle$) and subspaces T_k of N_J such that $P_{\leq k} = \{C \in P : \dim(C) \leq k\}$ is a cone decomposition of $T_k \otimes \mathbb{K}[x_1, \dots, x_k]$ and $T_k \subset \mathbb{K}[x_{k+1}, \dots, x_n]$ has a monomial basis for all $k = 1, \dots, r$. Furthermore $a_k \leq \frac{1}{2}a_{k+1}^2$ for $k = 1, \dots, r-1$.*

PROOF. We construct P inductively. Let $P_{>k} = \{C \in P : \dim(C) > k\}$ and consider $k = r$. Since P cannot contain cones with dimension greater than r , $P_{>r} = \emptyset$ and $P_{\leq r}$ is a cone decomposition of $N_J = T_r \otimes \mathbb{K}[x_1, \dots, x_r]$ with the monomial basis given in (3).

Now we assume that all cones of $P_{>k}$ have been constructed and that we already chose T_k such that

$$N_J = T_k \otimes \mathbb{K}[x_1, \dots, x_k] \oplus \bigoplus_{C \in P_{>k}} C.$$

We want to construct $P_{\leq k}$ inductively such that it is a cone decomposition of $T_k \otimes \mathbb{K}[x_1, \dots, x_k]$. By Lemma 3.16, P must contain exactly $\dim_{\mathbb{K}}(T_k)$ cones of dimension k . $P_{>k}$ is already constructed, so that $a_n, \dots, a_{k+1}$ are fixed. Since P shall be d_1 -exact, the cones of dimension k must have the degrees $a_{k+1}, a_{k+1} + 1, a_{k+1} + 2, \dots$. Let $\{t_1, \dots, t_l\}$ be a monomial basis of T_k with $\deg(t_1) \leq \dots \leq \deg(t_l)$. Then we choose

$$C_i = t_i x_k^{a_{k+1} + i - \deg(t_i) - 1} \mathbb{K}[x_1, \dots, x_k] \text{ with } i = 1, \dots, l$$

as cones of dimension k . It is easy to see that $\deg(C_i) = a_{k+1} + i - 1$ and $\dim(C_i) = k$. Thus we do not violate

the definition of exact cone decompositions. Since $T_k \subset \mathbb{K}[x_{k+1}, \dots, x_n]$, furthermore

$$T_k \otimes \mathbb{K}[x_1, \dots, x_k] = C_1 \oplus \dots \oplus C_l \oplus (T_{k-1} \otimes \mathbb{K}[x_1, \dots, x_{k-1}])$$

with

$$T_{k-1} = \text{span}_{\mathbb{K}} \{t_i x_k^e : i = 1, \dots, l, \\ e = 0, \dots, a_{k+1} + i - \deg(t_i) - 2\} \subset \mathbb{K}[x_k, \dots, x_n].$$

Inductively, this yields

$$N_J = (T_{k-1} \otimes \mathbb{K}[x_1, \dots, x_{k-1}]) \oplus \bigoplus_{C \in P_{>k-1}} C.$$

So it only remains to bound a_{k-1} .

$$a_{k-1} - a_k = \dim_{\mathbb{K}}(T_{k-1}) = \sum_{i=1}^l (a_{k+1} + i - \deg(t_i) - 1) \\ \leq \sum_{i=1}^l (a_{k+1} + i - 1) = la_{k+1} + \frac{1}{2}l(l-1)$$

With $l = \dim_{\mathbb{K}}(T_k) = a_k - a_{k+1}$, we get by induction

$$a_{k-1} \leq a_k + (a_k - a_{k+1})a_{k+1} + \frac{1}{2}(a_k - a_{k+1})(a_k - a_{k+1} - 1) \\ = \frac{1}{2}(a_k^2 - a_{k+1}^2 + a_k + a_{k+1}) \\ \leq \frac{1}{2}\left(a_k^2 - a_{k+1}^2 + \frac{1}{2}a_{k+1}^2 + a_{k+1}\right) \leq \frac{1}{2}a_k^2$$

□

COROLLARY 3.19. $a_k \leq 2 \left[\frac{1}{2}(d_1 \cdots d_{n-r} + d_1) \right]^{2^{r-k}}$ for $k = 1, \dots, r$.

Finally we remember that a_1 bounds the Gröbner basis degree and state our main theorem.

THEOREM 3.20. *Let $I = \langle f_1, \dots, f_s \rangle$ be an ideal in the ring $\mathbb{K}[X] = \mathbb{K}[x_1, \dots, x_n]$ generated by homogeneous polynomials of degrees $d_1 \geq \dots \geq d_s$. Then for any admissible ordering $\prec$, the degree required in a Gröbner basis for I with respect to $\prec$ is bounded by $2 \left[\frac{1}{2}(d_1 \cdots d_{n-r} + d_1) \right]^{2^{r-1}}$, where $r > 0$ is the (affine) dimension of I .*

PROOF. Corollary 3.19 gives a bound on a_1 . Since this bound is greater than $d_1 + \dots + d_{n-r} - n$, Corollary 3.13 and Proposition 3.4 finish the proof. □

Just like Dubé, we can lift this result to non-homogeneous ideals by introducing an additional homogenization variable x_{n+1} . This implies

COROLLARY 3.21. *Let $I = \langle f_1, \dots, f_s \rangle$ be an ideal in the ring $\mathbb{K}[X] = \mathbb{K}[x_1, \dots, x_n]$ generated by arbitrary polynomials of degrees $d_1 \geq \dots \geq d_s$. Then for any admissible ordering $\prec$, the degree required in a Gröbner basis for I with respect to $\prec$ is bounded by $2 \left(\frac{d_1^2}{2} + d_1 \right)^{2^{r-1}}$, where r is the dimension of I .*

If we consider an arbitrary non-trivial ideal, its dimension r is at most $n - 1$. For $r = n - 1$, the bound given in this paper simplifies to $2 \left(\frac{d_1^2}{2} + d_1 \right)^{2^{n-2}}$. This is exactly Dubé's bound in [5], Theorem 8.2.

However our bound bridges the gap to the case of zero-dimensional ideals. It is well-known that the Gröbner basis of I in this case can be at most the vector space dimension of $\mathbb{K}[X]/I$, which is bounded by $d_1 \cdots d_n$ according to the theorem of Bézout. Our bound (though not proved for $r = 0$) specializes to $d_1 \cdots d_n + d_1$ which is close to the perfect bound. For $0 < r < n - 1$ the bound is new to the best knowledge of the authors.

4. LOWER DEGREE BOUND

Finally we want to give a lower bound of similar form. Mayr, Meyer [13] and Möller, Mora [14] gave a lower bound for H-bases.

An H-basis of an ideal I is an ideal basis H such that $\langle \{h_{\deg(h)} : h \in H\} \rangle = \langle \{f_{\deg(f)} : f \in I\} \rangle$ (here $h_{\deg(h)}$ and $f_{\deg(f)}$ are the homogeneous components of highest degree). Consider a *graded* monomial ordering, i.e. $\deg(m) < \deg(m')$ implies $m \prec m'$ for all monomials m, m' . Then it is easy to see that any Gröbner basis with respect to $\prec$ is also an H-basis.

So we can reformulate the result as follows.

PROPOSITION 4.1 (MAYR, MEYER 1982).

There is a family of ideals $J_n \subset \mathbb{K}[X]$ with $n = 14(k + 1), k \in \mathbb{N}$, of polynomials in n variables of degree bounded by d such that each Gröbner basis with respect to a graded monomial ordering contains a polynomial of degree at least $\frac{1}{2}d^{2^{\frac{n}{14}-1}} + 4$.

We are going to embed this ideal in a larger ring as follows. Define

$$J_{r,n} = \langle J_r, x_{r+1}, \dots, x_n \rangle \subset \mathbb{K}[X].$$

Obviously $\dim(J_{r,n}) < r$.

THEOREM 4.2. *There is a family of ideals $J_{r,n} \subset \mathbb{K}[X]$ with $r = 14(k+1) \leq n, k \in \mathbb{N}$ of polynomials in n variables of degree bounded by d with dimension less than r such that each Gröbner basis with respect to a graded monomial ordering contains a polynomial of degree at least $\frac{1}{2}d^{2^{\frac{r}{14}-1}} + 4$.*

The constant $\frac{1}{14}$ in the exponent could be improved by applying the techniques of [14] to the improved construction in [17]. Furthermore it would be interesting to give a non-trivial upper bound on the dimension of the ideals J_n (resp. $J_{r,n}$). To the best of the authors' knowledge, only the lower bound $\dim(J_n) \geq \frac{3}{14}n + 12$ (cf. [14]) is known.

5. REFERENCES

- [1] D. Bayer. The division algorithm and the Hilbert scheme. 1982.
- [2] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, Universität Innsbruck, 1965.
- [3] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer New York, 1992.
- [4] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer Verlag, 2005.
- [5] T. Dubé. The Structure of Polynomial Ideals and Gröbner Bases. *SIAM Journal on Computing*, 19:750, 1990.

- [6] D. Eisenbud. *Commutative algebra with a view toward algebraic geometry*. Springer, 1995.
- [7] M. Giusti. Some effectivity problems in polynomial ideal theory. In *Eurosam*, volume 84, pages 159–171. Springer, 1984.
- [8] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.1.0 — A computer algebra system for polynomial computations. 2009.
<http://www.singular.uni-kl.de>.
- [9] M. Kratzer. *Computing the dimension of a polynomial ideal and membership in low-dimensional ideals*. Bachelor’s thesis, Technische Universität München, 2008.
- [10] M. Kreuzer and L. Robbiano. Computational commutative algebra 2. 2005.
- [11] K. Kühnle and E. Mayr. Exponential space computation of Gröbner bases. In *Proceedings of the 1996 international symposium on Symbolic and algebraic computation*, pages 63–71. ACM New York, NY, USA, 1996.
- [12] D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *Proc. EUROCAL*, volume 83, pages 146–156. Springer, 1983.
- [13] E. Mayr and A. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46(3):305–329, 1982.
- [14] H. Möller and F. Mora. *Upper and Lower Bounds for the Degree of Groebner Bases*. Springer-Verlag London, UK, 1984.
- [15] J. Schmid. On the affine Bezout inequality. *manuscripta mathematica*, 88(1):225–232, 1995.
- [16] I. Shafarevich. *Basic Algebraic Geometry*. Springer-Verlag, 1994.
- [17] C. Yap. A new lower bound construction for commutative Thue systems with applications. *J. Symbolic Comput.*, 12(1), 1991.

A New Algorithm for Computing Comprehensive Gröbner Systems *

Deepak Kapur
Dept. of Computer Science
University of New Mexico
Albuquerque, NM, USA
kapur@cs.unm.edu

Yao Sun
Key Laboratory of
Mathematics Mechanization
Academy of Mathematics and
Systems Science, CAS
Beijing, China
sunyao@amss.ac.cn

Dingkang Wang
Key Laboratory of
Mathematics Mechanization
Academy of Mathematics and
Systems Science, CAS
Beijing, China
dwang@mmrc.iss.ac.cn

ABSTRACT

A new algorithm for computing a comprehensive Gröbner system of a parametric polynomial ideal over $k[U][X]$ is presented. This algorithm generates fewer branches (segments) compared to Suzuki and Sato's algorithm as well as Nabeshima's algorithm, resulting in considerable efficiency. As a result, the algorithm is able to compute comprehensive Gröbner systems of parametric polynomial ideals arising from applications which have been beyond the reach of other well known algorithms. The starting point of the new algorithm is Weispfenning's algorithm with a key insight by Suzuki and Sato who proposed computing first a Gröbner basis of an ideal over $k[U, X]$ before performing any branches based on parametric constraints. Based on Kalkbrenner's results about stability and specialization of Gröbner basis of ideals, the proposed algorithm exploits the result that along any branch in a tree corresponding to a comprehensive Gröbner system, it is only necessary to consider one polynomial for each nondivisible leading power product in $k(U)[X]$ with the condition that the product of their leading coefficients is not 0; other branches correspond to the cases where this product is 0. In addition, for dealing with a disequality parametric constraint, a probabilistic check is employed for radical membership test of an ideal of parametric constraints. This is in contrast to a general expensive check based on Rabinovitch's trick using a new variable as in Nabeshima's algorithm. The proposed algorithm has been implemented in Magma and experimented with a number of examples from different applications. Its performance (vis a vie number of branches and execution timings) has been compared with the Suzuki-Sato's algorithm and Nabeshima's speed-up algorithm. The algorithm has been successfully used to solve the famous P3P problem from computer vision.

*The first author is supported by the National Science Foundation award CCF-0729097 and the last two authors are supported by NSFC 10971217, 10771206 60821002/F02.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]

General Terms

Algorithms

Keywords

Gröbner basis, comprehensive Gröbner system, radical ideal membership, probabilistic check.

1. INTRODUCTION

A new algorithm for computing a comprehensive Gröbner system (CGS), as defined by Weispfenning [18] for parametric ideals (see also [7] where a related concept of parametric Gröbner system was introduced) is proposed. The main advantage of the proposed algorithm is that it generates fewer branches (segments) compared to other related algorithms; as a result, the algorithm is able to compute comprehensive Gröbner systems for many problems from different application domains which could not be done previously. In the rest of this section, we provide some motivations for comprehensive Gröbner systems and approaches used for computing them.

Many engineering problems are parameterized and have to be repeatedly solved for different values of parameters [4]. A case in point is the problem of finding solutions of a parameterized polynomial system. One is interested in finding for what parameter values, the polynomial system has a common solution; more specifically, if there are solutions, one is also interested in finding out the structure of the solution space (finitely many, infinitely many, in which their dimension, etc.). One recent application of comprehensive Gröbner systems is in automated geometry theorem proving [2] and automated geometry theorem discovery [11]. In the former, the goal is to consider all possible cases arising from an ambiguous problem formulation to determine whether the conjecture is generic enough to be valid in all cases, or certain cases have to be ruled out. In the latter, one is interested in identifying different relationship among geometric entities for different parameter values. Another recent application is in the automatic generation of loop invariants and inductive assertions of programs operating on numbers using quantifier elimination methods as proposed in [8]. The main idea is to hypothesize invariants/assertions to have a template like structure (such as a polynomial in which

the degree of every variable is ≤ 2 , or a polynomial with a predetermined support), in which the presence/coefficient of a power product is parameterized. Verification conditions from the program are then generated which are formulas involving parameterized polynomial equations. The objective is to generate conditions on parameters which make these verification conditions to be valid. See [8] for more details.

Let k be a field, R be the polynomial ring $k[U]$ in the parameters $U = \{u_1, \dots, u_m\}$, and $R[X]$ be the polynomial ring over the parameter ring R in the variables $X = \{x_1, \dots, x_n\}$ and $X \cap U = \emptyset$, i.e., X and U are disjoint sets.

Given a polynomial set $F \subset R[X]$, we are interested in identifying conditions on parameters U such that the solution structure of the specialized polynomial system F for the values of U satisfying these conditions is different from other parameter values. One way to do this is to compute a comprehensive Gröbner system as introduced by Weispfenning, which is a finite set of triples of the form (E_i, N_i, G_i) , where E_i, N_i are finite sets of polynomials in $k[U]$ and $\sigma(G_i)$ is a Gröbner basis of $\sigma(F)$, for every specialization σ such that for every $e_i \in E_i$, e_i vanishes and for at least one $n_i \in N_i$, n_i does not vanish; we will say that in that case σ satisfies the parametric constraints. Furthermore, for every specialization, there is at least one triple whose parametric constraints satisfy it. We will call each triple as a *branch* (also called a segment) in a comprehensive Gröbner system.

In 1992, Weispfenning [18] gave an algorithm for computing a comprehensive Gröbner system but it suffered from the problem of too many branches, many of which leading to the Gröbner basis $\{1\}$.¹ Since then, many improvements have been made to improve these algorithms to make them useful for different applications; see [10, 14, 15, 9]. A major breakthrough was an algorithm proposed by Suzuki and Sato [16] (henceforth called the *SS* algorithm) in which they showed how traditional implementations of Gröbner basis algorithms for polynomial rings over a field could be exploited for computing a comprehensive Gröbner basis system.

The main idea of the *SS* algorithm is to compute a Gröbner basis G from the parametric ideal basis in $k[U, X]$ using the block ordering in which $U \ll X$. In case G has polynomials purely in the parameters U , there are branches corresponding to each such polynomial being not equal to 0 in which case the Gröbner basis is $\{1\}$ for the specialization. For the branch when all these polynomials are 0, the Gröbner basis is G minus these polynomials under the additional condition that the leading coefficient of each polynomial is nonzero. In addition, there are branches corresponding to the cases when each of these leading coefficients is 0.

Nabeshima's speed-up algorithm [12] improves upon the *SS* algorithm by using the fact that (i) for every leading power product, only one coefficient needs to be made nonzero, and (ii) Rabinovitch's trick of introducing a new variable can be used to make that polynomial monic. Nabeshima reported that these tricks led to fewer branches of the *SS*-algorithm for most examples.

The algorithm proposed in this paper uses ideas from the construction proposed by Weispfenning[19] for computing a canonical comprehensive Gröbner basis of a parametric ideal as the starting point. The proposed algorithm integrates the ideas about essential and inessential specializations from Weispfenning's construction with the key insight

¹Kapur's algorithm for parametric Gröbner bases suffered from similar weaknesses.

in the Suzuki-Sato (*SS*) algorithm based on Kalkbrener's results about specialization of ideals and stability of their Gröbner bases.

First, let G be the reduced Gröbner basis of a parametric ideal $\langle F \rangle \subset k[U, X]$ w.r.t. $\prec_{X,U}$, and let $G_r = G \cap k[U]$, the polynomials in parameters only in G . A noncomparable set G_m , which is defined in section 4, is extracted from $G \setminus G_r$, consisting only of polynomials with nondivisible powerproducts in X in G . Let h be the product of the leading coefficients of the polynomials in G_m . $(G_r, \{h\}, G_m)$ is one of the branches of the comprehensive Gröbner system of F . Based on case analysis over the leading coefficients of the polynomials in G_m , it is possible to compute the remaining branches of a comprehensive Gröbner system.

For computing a Gröbner basis for specializations along many branches, it is useful to perform radical membership check of a parametric constraint in an ideal of other parametric constraints for checking consistency. Instead of using Rabinovitch's trick of introducing a new variable for radical membership check as proposed in Nabeshima's speed-up version of the *SS* algorithm, we have developed a collection of useful heuristics for this check based on case analysis on whether the ideal whose radical membership is being checked, is 0-dimensional or not. In case of a positive dimensional ideal, a probabilistic check is employed after randomly specializing the independent variables of the ideal. The general check is performed as a last resort.

The paper is organized as follows. Section 2 gives notations and definitions used. Section 3 briefly reviews the Suzuki-Sato algorithm. Section 4 is the discussion of the key insights needed for the proposed algorithm; the new algorithm is presented there as well. Section 5 discusses heuristics for checking radical membership of an ideal. Section 6 illustrates the proposed algorithm on a simple example. Empirical data and comparison with the *SS*-algorithm and Nabeshima's speed-up algorithm are presented in Section 7. Concluding remarks follow in Section 8.

2. NOTATIONS AND DEFINITIONS

Let k be a field, R be the polynomial ring $k[U]$ in the parameters $U = \{u_1, \dots, u_m\}$, and $R[X]$ be the polynomial ring over R in the variables $X = \{x_1, \dots, x_n\}$ and $X \cap U = \emptyset$.

Let $PP(X)$, $PP(U)$ and $PP(U, X)$ be the sets of power products of X , U and $U \cup X$ respectively. $\prec_{X,U}$ is an admissible block term order on $PP(U, X)$ such that $U \ll X$. $\prec_X$ and $\prec_U$ is the restriction of $\prec_{X,U}$ on $PP(X)$ and $PP(U)$, respectively.

For a polynomial $f \in R[X] = k[U][X]$, the leading power product, leading coefficient and leading monomial of f w.r.t. the order $\prec_X$ are denoted by $\text{lpp}_X(f)$, $\text{lc}_X(f)$ and $\text{lm}_X(f)$ respectively. Since f can also be regarded as an element of $k[U, X]$, in this case, the leading power product, leading coefficient and leading monomial of f w.r.t. the order $\prec_{X,U}$ are denoted by $\text{lpp}_{X,U}(f)$, $\text{lc}_{X,U}(f)$ and $\text{lm}_{X,U}(f)$ respectively.

Given a field L , a specialization of R is a homomorphism $\sigma : R \rightarrow L$. In this paper, we assume L to be the algebraic closure of k , and consider the specializations induced by the elements in L^m . That is, for $\bar{a} \in L^m$, the induced homomorphism $\sigma_{\bar{a}}$ is denoted as $\sigma_{\bar{a}} : f \rightarrow f(\bar{a})$, $f \in R$. Every specialization $\sigma : R \rightarrow L$ extends canonically to a homomorphism $\sigma : R[X] \rightarrow L[X]$ by applying σ coefficient-wise.

DEFINITION 2.1. Let F be a subset of $R[X]$, $A_1, \dots, A_l$

be algebraically constructible subsets of L^m and $G_1, \dots, G_l$ be subsets of $R[X]$, and S be a subset of L^m such that $S \subseteq A_1 \cup \dots \cup A_l$. A finite set $\mathcal{G} = \{(A_1, G_1), \dots, (A_l, G_l)\}$ is called a *comprehensive Gröbner system* on S for F if $\sigma_{\bar{a}}(G_i)$ is a Gröbner basis of the ideal $\langle \sigma_{\bar{a}}(F) \rangle \subset L[X]$ for $\bar{a} \in A_i$ and $i = 1, \dots, l$. Each (A_i, G_i) is called a *branch* of $\mathcal{G}$. If $S = L^m$, $\mathcal{G}$ is called a *comprehensive Gröbner system* for F .

DEFINITION 2.2. A comprehensive Gröbner system $\mathcal{G} = \{(A_1, G_1), \dots, (A_l, G_l)\}$ on S for F is said to be *minimal* if for every $i = 1, \dots, l$, (i) for each $g \in G_i$, $\sigma_{\bar{a}}(\text{lc}_X(g)) \neq 0$ for any $\bar{a} \in A_i$, (ii) $\sigma_{\bar{a}}(G_i)$ is a minimal Gröbner basis of the ideal $\langle \sigma_{\bar{a}}(F) \rangle \subset L[X]$ for $\bar{a} \in A_i$, and (iii) $A_i \neq \emptyset$, and furthermore, for each $i, j = 1 \dots l$, $A_i \cap A_j = \emptyset$ whenever $i \neq j$.

For an $F \subset R = k[U]$, the variety defined by F in L^m is denoted by $V(F)$. In this paper, the constructible set A_i always has the form: $A_i = V(E_i) \setminus V(N_i)$ where E_i, N_i are subsets of $k[U]$. If $A_i = V(E_i) \setminus V(N_i)$ is empty, the branch (A_i, G_i) is redundant.

DEFINITION 2.3. For $E, N \subset R = k[U]$, a pair (E, N) is called a *parametric constraint*. A parametric constraint (E, N) is said to be *consistent* if the set $V(E) \setminus V(N)$ is not empty. Otherwise, (E, N) is called *inconsistent*.

It is easy to see that the consistency of (E, N) can be checked by ensuring that at least one $f \in N$ is not in the radical of $\langle E \rangle$.

3. THE SUZUKI-SATO ALGORITHM

In this section, we briefly review the key ideas of the Suzuki-Sato algorithm [16]. The following two lemmas serve as the basis of the SS algorithm. The first lemma is a corollary of the Theorem 3.1 given by Kalkbrener in [6].

LEMMA 3.1. Let G be a Gröbner basis of the ideal $\langle F \rangle \subset k[U, X]$ w.r.t. the order $\prec_{X,U}$. For any $\bar{a} \in L^m$, let $G_1 = \{g \in G \mid \sigma_{\bar{a}}(\text{lc}_X(g)) \neq 0\}$. Then $\sigma_{\bar{a}}(G_1) = \{\sigma_{\bar{a}}(g) \mid g \in G_1\}$ is a Gröbner basis of $\langle \sigma_{\bar{a}}(F) \rangle$ in $L[X]$ w.r.t. $\prec_X$ if and only if $\sigma_{\bar{a}}(g)$ reduces to 0 modulo $\sigma_{\bar{a}}(G_1)$ for every $g \in G$.

The next lemma, which follows from the first lemma, plays the key role in the design of the SS algorithm.

LEMMA 3.2. Let G be a Gröbner basis of the ideal $\langle F \rangle \subset k[U, X]$ w.r.t. the order $\prec_{X,U}$. If $\sigma_{\bar{a}}(\text{lc}_X(g)) \neq 0$ for each $g \in G \setminus (G \cap R)$, then $\sigma_{\bar{a}}(G)$ is a Gröbner basis of $\langle \sigma_{\bar{a}}(F) \rangle$ in $L[X]$ w.r.t. $\prec_X$ for any $\bar{a} \in V(G \cap R)$.

The main idea of the SS algorithm is to first compute a reduced Gröbner basis, say G , of $\langle F \rangle \subset k[U, X]$ w.r.t. $\prec_{X,U}$, which is also a Gröbner basis of the ideal $\langle F \rangle \subset k[U][X]$ w.r.t. $\prec_X$. Let $\{h_1, \dots, h_l\} = \{\text{lc}_X(g) \mid g \in G \setminus R\} \subset R$. By the above lemma, $(G \cap k[U], V(h_1) \cup \dots \cup V(h_l), G)$ forms a branch of the comprehensive Gröbner system for F . That is, for any $\bar{a} \in V(G \cap k[U]) \setminus (V(h_1) \cup \dots \cup V(h_l))$, $\sigma_{\bar{a}}(G)$ is a Gröbner basis of $\langle \sigma_{\bar{a}}(F) \rangle$ in $L[X]$ w.r.t. $\prec_X$. To compute other branches corresponding to the specialization $\bar{a} \in V(h_1) \cup \dots \cup V(h_l)$, Lemma 3.2 is used for each $F \cup \{h_i\}$, the above steps are repeated. Since $h_i \notin \langle F \rangle$, the algorithm terminates in finitely many steps.

As stated earlier, this algorithm can be easily implemented in most of the computer algebra systems already supporting

an efficient implementation of a Gröbner basis algorithm over a polynomial ring over a field. It has very good performance since it can take advantage of well-known fast implementations for computing Gröbner bases.

The algorithm however suffers from certain weaknesses. The algorithm does not check whether $V(G \cap R) \setminus V(h)$ is empty; as a result, many redundant/unnecessary branches may be produced. In [16], an improved version of the algorithm is reported which removes redundant branches. To reduce the number of branches generated from the SS algorithm, Nabeshima proposed a speed-up algorithm in [12]. The main idea of that algorithm is to exploit disequality parametric constraints for simplification. For every leading power product in $G \setminus R$ that is a nontrivial multiple of any other leading product in it, a branch is generated by asserting its leading coefficient h_i to be nonzero. The corresponding polynomial is made monic using Rabinovitch's trick of introducing a new variable to handle the disequality $h_i \neq 0$, and the Gröbner basis computation is performed again, simplifying polynomials whose leading power products are multiples, including their parametric coefficients.

4. THE PROPOSED ALGORITHM

We present below a new algorithm for computing a comprehensive Gröbner system which avoids unnecessary branches in the SS algorithm. This is done using the radical ideal membership check for parametric constraints asserted to be nonzero. Heuristics are employed to do this check; when these heuristics fail, as exhibited by Table 2 in Section 7 on experimental results, only then the general check is performed by introducing a new variable, since this check is very inefficient because of the extra variable. Further, all parametric constraints leading to the specialized Gröbner basis being 1 are output as a single branch, leading to a compactified output.

Another major improvement of the proposed algorithm is that along any other branch for which the specialized Gröbner basis is different from 1, exactly one polynomial from $G \setminus R$ per minimal leading power product is selected. This is based on a generalization of Kalkbrener's Theorem 3.1. All these results are integrated into the proposed algorithm, resulting in considerable efficiency over the SS algorithm and Nabeshima's improved algorithm by avoiding expensive Gröbner basis computations along most branches.

The proposed algorithm is based on the following theorem. The definitions below are used in the theorem.

DEFINITION 4.1. Given a set G of polynomials which are a subset of $k[U, X]$ and an admissible block order with $U \ll X$, let $\text{Noncomparable}(G)$ be a subset, called F , of G such that (i) for every polynomial $g \in G$, there is some polynomial $f \in F$ such that $\text{lpp}_X(g)$ is a multiple of $\text{lpp}_X(f)$ and (ii) for any two distinct $f_1, f_2 \in F$, neither $\text{lpp}_X(f_1)$ is a multiple of $\text{lpp}_X(f_2)$ nor $\text{lpp}_X(f_2)$ is a multiple of $\text{lpp}_X(f_1)$.

It is easy to see that $\langle \text{lpp}_X(\text{Noncomparable}(G)) \rangle = \langle \text{lpp}_X(G) \rangle$. The following simple example shows that $\text{Noncomparable}(G)$ may not be unique.

Let $G = \{ax^2 - y, ay^2 - 1, ax - 1, (a+1)x - y, (a+1)y - a\} \subset \mathbb{Q}[a, x, y]$, with the lexicographic order on terms with $a < y < x$. Then $F = \{ax - 1, (a+1)y - a\}$ and $F' = \{(a+1)x - y, (a+1)y - a\}$ are both $\text{Noncomparable}(G)$. It is easy to verify $\langle \text{lpp}_X(F) \rangle = \langle \text{lpp}_X(F') \rangle = \langle \text{lpp}_X(G) \rangle = \langle x, y \rangle$.

DEFINITION 4.2. Given $F \subset k[U, X]$ and $p \in k[U, X]$, p is said to be divisible by F if there exists an $f \in F$ such that some power product in X of p is divisible by $\text{lpp}_X(f)$.

THEOREM 4.3. Let G be a Gröbner basis of the ideal $\langle F \rangle \subset k[U, X]$ w.r.t. an admissible block order with $U \ll X$. Let $G_r = G \cap k[U]$ and $G_m = \text{Noncomparable}(G \setminus G_r)$. Denote $h = \prod_{g \in G_m} \text{lc}_X(g) \in k[U]$. If σ is a specialization from $k[U]$ to L such that $\sigma(g) = 0$ for $g \in G_r$ and $\sigma(h) \neq 0$, then $\sigma(G_m)$ is a Gröbner basis of $\langle \sigma(F) \rangle$ in $L[X]$ w.r.t. $\prec_X$.

PROOF. Consider any $p \in G \setminus (G_r \cup G_m)$; p is divisible by G_m . p can be transformed by multiplying it with the leading coefficients of polynomials in G_m and then reduced using G_m , and then this process can be repeated on the result. Let r be the remainder of p w.r.t. G_m in X obtained by multiplying p by the leading coefficient of $g \in G_m$ such that r does not have any power product that is a multiple of any of the leading power products of polynomials in G_m (r could be different depending upon the order in which different polynomials in G_m are used to transform p). Thus,

$$(\text{lc}_X(g_1))^{\alpha_1} \cdots (\text{lc}_X(g_s))^{\alpha_s} p = q_1 g_1 + \cdots + q_s g_s + r, \quad (1)$$

where $g_i \in G_m$, $q_i \in k[U, X]$ for $i = 1, \dots, s$, $r \in k[U, X]$ such that no power product of r in X is a multiple of any of the leading power products of G_m . Since $p \in \langle F \rangle$, $r \in \langle F \rangle$. Since G is a Gröbner basis of $\langle F \rangle$ in $k[U, X]$, r reduces to 0 by G . However, r is reduced (in normal form) w.r.t. G_m in X (and hence reduced w.r.t. $G \setminus G_r$ in X also, by the definition of G_m); so r reduces to 0 by G_r only and further no new power products in X can be introduced during the simplification of r by G_r . So $r \in \langle G_r \rangle \subset k[U, X]$. Additionally, $\text{lpp}_X(p) \geq \text{lpp}_X(q_i g_i)$ since $\text{lc}_X(g_i) \in k[U]$.

Let $c = (\text{lc}_X(g_1))^{\alpha_1} \cdots (\text{lc}_X(g_s))^{\alpha_s}$. Apply σ to the both sides of (1), then we have:

$$\sigma(c)\sigma(p) = \sigma(q_1)\sigma(g_1) + \cdots + \sigma(q_s)\sigma(g_s) + \sigma(r).$$

Since $\sigma(h) \neq 0$ by assumption, $\sigma(\text{lc}_X(g)) \neq 0$ for $g \in G_m$; $\sigma(g) = 0$ for $g \in G_r$ which implies that $\sigma(r) = 0$. Notice $0 \neq \sigma(c) \in L$ and $\text{lpp}_X(p) \geq \text{lpp}_X(q_i g_i)$, using the following lemma, $\sigma(G_m)$ is a Gröbner basis of $\langle \sigma(G) \rangle = \langle \sigma(F) \rangle$. $\square$

In the above theorem, if $G_r = \emptyset$, then G_m is actually a Gröbner basis of the ideal $\langle F \rangle \subset k(U)[X]$.

We assume that the reader is familiar with the concept of t -representations which is often used to determine if a set of polynomials is a Gröbner basis; for details, consult [1].

LEMMA 4.4. Let G be a Gröbner basis of $\langle G \rangle \subset k[U, X]$ w.r.t. an admissible block order with $U \ll X$. Let $G_1 = \{g_1, \dots, g_s\} \subset G$ and σ be a specialization from $k[U]$ to L such that $\sigma(\text{lc}_X(g_i)) \neq 0$ for $i = 1, \dots, s$. If for each $p \in G \setminus G_1$, there exist $p_1, \dots, p_s \in L[X]$ such that: $\sigma(p) = p_1 \sigma(g_1) + \cdots + p_s \sigma(g_s)$, where $\text{lpp}_X(p) \geq \text{lpp}_X(p_i \sigma(g_i))$ for $i = 1, \dots, s$, then $\sigma(G_1)$ is a Gröbner basis of $\langle \sigma(G) \rangle$ in $L[X]$ w.r.t. $\prec_X$.

PROOF. By the hypothesis, it is easy to check $\sigma(G) \subset \langle \sigma(G_1) \rangle$ and hence $\sigma(G_1)$ is a basis of $\langle \sigma(G) \rangle$. So it remains to show $\sigma(G_1)$ is a Gröbner basis.

For each $g_j, g_k \in G_1$, we compute the s-polynomial of $\sigma(g_j)$ and $\sigma(g_k)$ in $L[X]$. Since $\sigma(\text{lc}_X(g_j)) \neq 0$ and $\sigma(\text{lc}_X(g_k)) \neq 0$, we have

$$\text{spoly}(\sigma(g_j), \sigma(g_k)) = c\sigma(\text{spoly}_X(g_j, g_k)), \quad (2)$$

where c is a nonzero constant in L and $\text{spoly}_X(g_j, g_k) \in k[U][X]$ is the s-polynomial of g_j and g_k w.r.t. X .

Assume $G \setminus G_1 = \{g_{s+1}, \dots, g_l\}$. Since G is a Gröbner basis of $\langle G \rangle \subset k[U, X]$ and $\text{spoly}_X(g_j, g_k) \in \langle G \rangle \subset k[U, X]$, there exist $h_1, \dots, h_l \in k[U, X]$ such that

$$\text{spoly}_X(g_j, g_k) = h_1 g_1 + \cdots + h_l g_l,$$

where $\text{lcm}(\text{lpp}_X(g_j), \text{lpp}_X(g_k)) \succ \text{lpp}_X(h_i g_i)$ for $i = 1, \dots, l$. Substitute back to (2), then obtain:

$$\text{spoly}(\sigma(g_j), \sigma(g_k)) = c(\sigma(h_1)\sigma(g_1) + \cdots + \sigma(h_l)\sigma(g_l)), \quad (3)$$

where $\text{lcm}(\text{lpp}_X(\sigma(g_j)), \text{lpp}_X(\sigma(g_k))) = \text{lcm}(\text{lpp}_X(g_j), \text{lpp}_X(g_k)) \succ \text{lpp}_X(h_i g_i) \geq \text{lpp}_X(\sigma(h_i))\text{lpp}_X(g_i)$ for $i = 1, \dots, l$. The next step is to use the hypothesis that for each $p \in G \setminus G_1$, there exist $p_1, \dots, p_s \in L[X]$ such that: $\sigma(p) = p_1 \sigma(g_1) + \cdots + p_s \sigma(g_s)$, where $\text{lpp}_X(p) \geq \text{lpp}_X(p_i \sigma(g_i))$ for $i = 1, \dots, s$. Substitute these representations back to (3), we get

$$\text{spoly}(\sigma(g_j), \sigma(g_k)) = p'_1 \sigma(g_1) + \cdots + p'_s \sigma(g_s), \quad (4)$$

where $p'_1, \dots, p'_s \in L[X]$ and $\text{lcm}(\text{lpp}_X(\sigma(g_j)), \text{lpp}_X(\sigma(g_k))) \succ \text{lpp}_X(p'_i \sigma(g_i))$ for $i = 1, \dots, s$. In fact, (4) is a t -representation of $\text{spoly}(\sigma(g_j), \sigma(g_k))$ with $t \prec \text{lcm}(\text{lpp}_X(\sigma(g_j)), \text{lpp}_X(\sigma(g_k)))$. Therefore, by the theory of t -representations, $\sigma(G_1)$ is a Gröbner basis. The lemma is proved. $\square$

4.1 Algorithm

We are now ready to give the algorithm for computing a minimal comprehensive Gröbner system. Its proof of correctness uses Theorem 4.3. Its termination can be proved in a way similar to the SS algorithm presented in [16].

In order to keep the presentation simple so that the correctness and termination of the algorithm are evident, we have deliberately avoided tricks and optimizations such as factoring h below. All the tricks suggested in the SS algorithm can be used here as well. In fact, our implementation incorporates fully these optimizations.

Algorithm PGBMain

Input: (E, N, F) : E, N , finite subsets of $k[U]$; F , a finite subset of $k[U, X]$.

Output: a finite set of 3-tuples (E_i, N_i, G_i) such that $\{(V(E_i) \setminus V(N_i), G_i)\}$ constitute a minimal comprehensive Gröbner system of F on $V(E) \setminus V(N)$.

begin

if $V(E) \setminus V(N) = \emptyset$ **then return** $\emptyset$ **end if**

$G \leftarrow \text{ReducedGröbnerBasis}(F \cup E, \prec_{X,U})$

if $1 \in G$ **then return** $\{(E, N, \{1\})\}$ **end if**

$G_r \leftarrow G \cap k[U] \quad \# V(G_r) \subset V(E)$

if $(V(E) \setminus V(G_r)) \setminus V(N) = \emptyset$

then $\mathcal{PGB} \leftarrow \emptyset$

else $\mathcal{PGB} \leftarrow \{(E, G_r \wedge N, \{1\})\}$

end if

if $V(G_r) \setminus V(N) = \emptyset$

then return $\mathcal{PGB}$;

else $G_m \leftarrow \text{Noncomparable}(G \setminus G_r)$

$\{h_1, \dots, h_s\} \leftarrow \{\text{lc}_X(g) : g \in G_m\}$

$h \leftarrow \text{lcm}\{h_1, \dots, h_s\}$;

if $(V(G_r) \setminus V(N)) \setminus V(h) \neq \emptyset$ **then**

$\mathcal{PGB} \leftarrow \mathcal{PGB} \cup \{(G_r, N \wedge \{h\}, G_m)\}$

end if

$\mathcal{PGB} \leftarrow \mathcal{PGB} \cup \mathcal{PGBMain}(G_r \cup \{h_1\}, N, G \setminus G_r) \cup$

$\mathcal{PGBMain}(G_r \cup \{h_2\}, N \wedge \{h_1\}, G \setminus G_r) \cup$

$\mathcal{PGBMain}(G_r \cup \{h_3\}, N \wedge \{h_1 h_2\}, G \setminus G_r) \cup$

$\dots$

PGBMain($G_r \cup \{h_s\}, N \wedge \{h_1 \cdots h_{s-1}\}, G \setminus G_r$)
return PGB

end if
end

In the above algorithm, $A \wedge B = \{fg | f \in A, g \in B\}$. Checking whether $V(A) \setminus V(B)$ is empty, is equivalent to the inconsistency of the parametric constraint (A, B) . Similarly checking whether $(V(A) \setminus V(B)) \setminus V(C) = V(A) \setminus (V(B) \cup V(C))$ is empty, is equivalent to checking whether $(A, B \wedge C)$ is inconsistent. The next section focuses on how the consistency check of a parametric constraint is performed.

As should be evident, a branch is never generated for the case when (E_i, N_i) is inconsistent. Further, the constructible sets are disjoint by construction. More importantly, branching is done only based on the leading coefficients of $G_m = \text{Noncomparable}(G \setminus G_r)$, instead of the whole $G \setminus G_r$. As a result, the number of branches generated by the above algorithm is strictly smaller than that of the branches in the SS algorithm. In addition, efficient heuristics are employed to perform the consistency check; as a last resort only when other heuristics do not work, we introduce a new variable to do the consistency check. In fact, this general check is rarely performed as confirmed by experimental data discussed in Section 7. Because of these optimizations, the proposed algorithm has a much better performance than the SS algorithm as well as Nabeshima's speed-up algorithm, as experimentally shown in Section 7.

As shown in [16], a comprehensive Gröbner basis can be computed by adapting the above algorithm for computing a comprehensive Gröbner system by using a new variable. The same technique can be applied to the above algorithm as well for computing a comprehensive Gröbner basis.

5. CONSISTENCY OF PARAMETRIC CONSTRAINTS

As should be evident from the above description of the algorithm, there are two main computational steps which are being repeatedly performed: (i) Gröbner basis computations, and (ii) checking consistency of parametric constraints. As stated above, a parametric constraint (E, N) , $E, N \subset k[U]$ is inconsistent if and only if for each $f \in N$, f is in the radical ideal of $\langle E \rangle$. This section discusses heuristics we have integrated into the implementation of the algorithm for the check whether $(E, \{f\})$ is inconsistent. In this section, we always assume that E itself is a Gröbner basis.

A general method to check whether $f \in \sqrt{\langle E \rangle}$ is to introduce a new variable y and compute the Gröbner basis G_y of $\langle E \cup \{fy - 1\} \rangle \subset k[U, y]$ for any admissible monomial order. If $G_y = \{1\}$, then $f \in \sqrt{\langle E \rangle}$ and $(E, \{f\})$ is inconsistent. Otherwise, $(E, \{f\})$ is consistent. However, this method can be, in general, very expensive partly because of introduction of a new variable. Consequently, this method is used only as a last resort when other heuristics fail.

The first heuristic is to check whether f is in the ideal generated by E ; since in the algorithm, a Gröbner basis of E is already available, the normal form of f is computed; if it is 0, then f is in the ideal of E implying that $(E, \{f\})$ is inconsistent. This heuristic turns out to be quite effective as shown from experimental results in Section 7.

Otherwise, different heuristics are used depending upon whether E is 0-dimensional or not. In case E is 0-dimensional, the method discussed in the next subsection for the radical

membership check is complete, i.e., it decides whether f is in the radical ideal of E or not. In case E is of positive dimension, then roughly, independent variables are assigned randomly, hopefully, resulting in a 0-dimensional ideal, for which the radical membership check can be done. However, this heuristic is not complete. If this heuristic cannot determine whether $(E, \{f\})$ is inconsistent, then another heuristic is employed that checks whether f^{2^k} is in the ideal of E for a suitably small value of k .

5.1 Ideal(E) is 0-dimensional

For the case when E is 0-dimensional, linear algebra techniques can be used to check the radical membership in E . The main idea is to compute the characteristic polynomial of the linear map associated with f , which can be efficiently done using a Gröbner basis of E .

Let $A = k[U]/\langle E \rangle$. Consider the map induced by $f \in k[U]$: $m_f : A \rightarrow A$, $[g] \mapsto [fg]$, where $g \in k[U]$ and $[g]$ is its equivalence class in A .

See [3, 17] for the proofs of the following lemmas.

LEMMA 5.1. *Assume that the map m_f is defined as above. Then,*

- (1) m_f is the zero map exactly when $f \in \langle E \rangle$.
- (2) For a univariate polynomial q over k , $m_{q(f)} = q(m_f)$.
- (3) $p_f(f) \in \langle E \rangle$, where p_f is the characteristic polynomial of m_f .

LEMMA 5.2. *Let $p_f \in k[\lambda]$ be the characteristic polynomial of m_f . Then for $\alpha \in L$, the following statements are equivalent.*

- (1) α is a root of the equation $p_f(\lambda) = 0$.
- (2) α is a value of the function f on $V(E)$.

Using these lemmas, we have:

PROPOSITION 5.3. *Let $p_f \in k[\lambda]$ be the characteristic polynomial of m_f and $d = \deg(p_f)$.*

- (1) $p_f = \lambda^d$ if and only if $f \in \sqrt{\langle E \rangle}$.
- (2) $p_f = q$ and $\lambda \nmid q$ if and only if there exists $g \in k[U]$ such that $gf \equiv 1 \pmod{\langle E \rangle}$.
- (3) $p_f = \lambda^{d'} q$, where $0 < d' < d$ and $\lambda \nmid q$ if and only if $f \notin \sqrt{\langle E \rangle}$ and there exists $g \notin \sqrt{\langle E \rangle}$ such that $fg \in \sqrt{\langle E \rangle}$.

PROOF. (1) $\Rightarrow$ If $p_f = \lambda^d$, then $p_f(f) = f^d \in \langle E \rangle$ by lemma 5.1, which shows $f \in \sqrt{\langle E \rangle}$. $\Leftarrow$ Since $f \in \sqrt{\langle E \rangle}$, 0 is the sole value of the function f on $V(E)$. By lemma 5.2, $p_f = \lambda^d$.

(2) $\Rightarrow$ If $p_f = q$ and $\lambda \nmid q$, then there exist $a, b \in k[\lambda]$ such that $a\lambda + bp_f = 1$. Substitute λ by f . Then obtain $a(f)f + b(f)p_f(f) = 1$. $p_f(f) \in \langle E \rangle$ shows $a(f)f \equiv 1 \pmod{\langle E \rangle}$. $\Leftarrow$ If there exists $g \in k[U]$ such that $gf \equiv 1 \pmod{\langle E \rangle}$, then all the values of the function f on $V(f)$ are not 0, which means the roots of $p_f(\lambda) = 0$ are not 0 as well by the above lemma. So $\lambda \nmid p_f$.

(3) $\Rightarrow$ If $p_f = \lambda^{d'} q$, where $0 < d' < d$ and $\lambda \nmid q$, then we have $f \notin \sqrt{\langle E \rangle}$ by (1). By lemma 5.1, $p_f(f) = f^{d'} q(f) \in \langle E \rangle$, and hence, $f q(f) \in \sqrt{\langle E \rangle}$. It remains to show $q(f) \notin \sqrt{\langle E \rangle}$. We prove this by contradict. If $q(f) \in \sqrt{\langle E \rangle}$, then there exists an integer $c > 0$ such that $q^c(f) \in \langle E \rangle$, which implies $m_{q^c(f)} = q^c(m_f) = 0$. Thus, q^c is a multiple of the minimal polynomial of m_f and hence all the irreducible factors of p_f should be factors of q^c . But this contradicts with $\lambda \nmid q$. $\Leftarrow$ Since $f, g \notin \sqrt{\langle E \rangle}$ and $fg \in \sqrt{\langle E \rangle}$, both

f and g are nonzero functions on $V(E)$, but fg is a zero function on $V(E)$. This implies that f vanishes on some but not all points of $V(E)$. By lemma 5.2, $p_f = \lambda^{d'} q$, where $0 < d' < d$ and $\lambda \nmid q$. $\square$

For the case (2) of proposition 5.3, clearly $V(E) \setminus V(f) = V(E)$ holds. For the case (3), it is easy to check $V(E) \setminus V(f) = V(E \cup \{q(f)\})$ by Lemma 5.2. So the parametric constraint $(E, \{f\})$ is equivalent to $(E \cup \{q(f)\}, \{1\})$, which converts the disequality constraint into equality constraint. Both (2) and (3) will speed up the implementation of the new algorithm.

If E is zero-dimensional, then $k[U]/\langle E \rangle$ is a finite vector space and the characteristic polynomial of m_f can be generated in [3]. Since in our algorithm, E itself is a Gröbner basis, the complexity of doing radical membership check is of polynomial time, which is much more efficient than the general method based on Rabinovitch's trick.

The following algorithm is based on the above theory:

Algorithm Zero-DimCheck

Input: $(E, \{f\})$: E is the Gröbner basis of the zero dimensional ideal $\langle E \rangle$; f , a polynomial in $k[U]$.

Output: **true** (consistent) or **false** (inconsistent).

begin

$p_f \leftarrow$ characteristic polynomial of m_f defined on $k[U]/\langle E \rangle$

$d \leftarrow \deg(p_f)$

if $p_f \neq \lambda^d$ **then return true else return false** **end if**

end

5.2 Ideal(E) is of positive dimension

We discuss two heuristics, *CCheck* and *ICheck*, for radical membership check; neither one is complete.

A subset V of U is *independent* modulo the ideal I if $k[V] \cap I = \{0\}$. An independent subset of U is maximal if there is no independent subset containing V properly.

The following proposition is well-known.

PROPOSITION 5.4. *Let $I \subset k[U]$ be an ideal and $\prec_U$ be a graded order on $k[U]$. If $k[V] \cap \text{lpp}_U(I) = \emptyset$, then $k[V] \cap I = \emptyset$. Furthermore, the maximal independent subset modulo $\text{lpp}_U(I)$ is also a maximal independent subset modulo I .*

A maximal independent subset modulo the monomial ideal of $\langle E \rangle$ can be easily computed; the above proposition thus provides a method to compute the maximal independent subset modulo an ideal.

The following theorem is obvious, so the proof is omitted.

THEOREM 5.5. *Let $\langle E \rangle \subset k[U]$ with positive dimension, V be a maximal independent subset modulo $\langle E \rangle$, and $\bar{\alpha}$ be an element in k^l where l is the cardinality of V . If $f|_{V=\bar{\alpha}} \notin \sqrt{\langle E|_{V=\bar{\alpha}} \rangle}$, then $f \notin \sqrt{\langle E \rangle}$ i.e. $(E, \{f\})$ is consistent.*

Since V is a maximal independent subset modulo $\langle E \rangle$, the ideal $\langle E \rangle$ becomes a zero dimensional ideal in $k[U \setminus V]$ with probability 1 by setting V to a value in k^l randomly when the characteristic of k is 0. In this case, we can use the technique provided in the last subsection to check if $f|_{V=\bar{\alpha}} \notin \sqrt{\langle E|_{V=\bar{\alpha}} \rangle}$. If $(E|_{V=\bar{\alpha}}, f|_{V=\bar{\alpha}})$ is consistent, then $(E, \{f\})$ is consistent. This gives an algorithm for checking the consistence of $(E, \{f\})$. When $f \notin \sqrt{\langle E \rangle}$, this algorithm can detect it efficiently.

Algorithm CCheck

Input: $(E, \{f\})$: E is the Gröbner basis of $\langle E \rangle$ w.r.t. a

graded monomial order $\prec_U$; f , a polynomial in $k[U]$.

Output: **true** (consistent) or **false**.

begin

$V \leftarrow$ independent variables of $\langle \text{lpp}_U(E) \rangle$

$\bar{\alpha} \leftarrow$ a random element in k^l

$spE \leftarrow \text{GröbnerBasis}(E|_{V=\bar{\alpha}}, \prec_U)$

if $\langle spE \rangle$ is zero dimension in $k[U \setminus V]$ **then**

if $\text{ZeroDimCheck}(spE, f|_{V=\bar{\alpha}}) = \text{true}$ **then**

return true

end if

end if;

return false

end

In the above algorithm, we only need to compute the Gröbner basis of $\langle E|_{V=\bar{\alpha}} \rangle$ which is usually zero dimensional and has fewer variables. So *CCheck* is more efficient than the general method which needs to compute the Gröbner basis of $\langle E \cup \{fy - 1\} \rangle$ whose dimension is positive.

If *CCheck*($E, \{f\}$) returns true, then $(E, \{f\})$ is consistent. However, if *CCheck*($E, \{f\}$) returns false, it need not be the case that $(E, \{f\})$ is inconsistent.

The following simple heuristic *ICheck* checks whether f^{2^k} is in the ideal generated by E by repeatedly squaring the normal form of f^{2^i} in an efficient way.

Algorithm ICheck

Input: $(E, \{f\})$: E is the Gröbner basis of $\langle E \rangle$ w.r.t. $\prec_U$; f , a polynomial in $k[U]$.

Output: **true** (inconsistent) or **false**.

begin

$loops \leftarrow$ an integer given in advance

$p \leftarrow f$

for i **from** 1 **to** $loops$ **do**

$\{m_1, \dots, m_l\} \leftarrow$ monomials of p

$s \leftarrow 0$

for $m \in \{m_1, \dots, m_l\}$ **do**

$s \leftarrow s + \text{NormalForm}(p \cdot m, E)$

end for

if $s = 0$ **then return true** **end if**

$p \leftarrow s$

end for

return false

end

Clearly, if *ICheck*($E, \{f\}$) returns true, then $(E, \{f\})$ is inconsistent.

5.3 Putting All Together

The above discussed checks are done in the following order for checking the consistency of a parametric constraint $(E, \{f\})$. First check whether f is in the ideal of E ; this check can be easily done by computing the normal form of f using a Gröbner basis of E which is readily available. If yes, then the constraint is inconsistent. If no, then depending upon the dimension of the ideal of E , either *Zero-DimCheck* or *CCheck* is performed. If E is 0-dimensional, then the check is complete in that it decides whether the constraint is consistent or not. If E is of positive dimension then if *CCheck* returns true, the constraint is consistent; otherwise, *ICheck* is performed. If *ICheck* succeeds, then the constraint is inconsistent. Finally, the general check is performed by computing a Gröbner basis of $E \cup \{fy - 1 = 0\}$, where y is a new variable different from U .

6. A SIMPLE EXAMPLE

The proposed algorithm is illustrated on an example.

EXAMPLE 6.1. Let $F = \{ax - b, by - a, cx^2 - y, cy^2 - x\} \subset \mathbb{Q}[a, b, c][x, y]$, with the block order $\prec_{X,U}, \{a, b, c\} \ll \{x, y\}$; within each block, $\prec_X$ and $\prec_U$ are graded reverse lexicographic orders with $y < x$ and $c < b < a$, respectively.

(1) We have $E = \emptyset$, $N = \{1\}$: the parametric constraint (E, N) is consistent. The reduced Gröbner basis of $\langle F \rangle$ w.r.t. $\prec_{X,U}$ is $G = \{x^3 - y^3, cx^2 - y, ay^2 - bc, cy^2 - x, ax - b, bx - acy, a^2y - b^2c, by - a, a^6 - b^6, a^3c - b^3, b^3c - a^3, ac^2 - a, bc^2 - b\}$; $G_r = G \cap \mathbb{Q}[a, b, c] = \{a^6 - b^6, a^3c - b^3, b^3c - a^3, ac^2 - a, bc^2 - b\}$. It is easy to see that (E, G_r) and $(E, G_r \wedge N)$ are consistent. This leads to the trivial branch of the comprehensive Gröbner system for F : $(\emptyset, G_r, \{1\})$.

(2) $G \setminus G_r = \{x^3 - y^3, cx^2 - y, ay^2 - bc, cy^2 - x, ax - b, bx - acy, a^2y - b^2c, by - a\}$; $G_m = \text{Noncomparable}(G \setminus G_r) = \{bx - acy, by - a\}$. Further, $h = \text{lcm}\{\text{lc}_X(bx - acy), \text{lc}_X(by - a)\} = b$. This results in another branch of the comprehensive Gröbner system for F corresponding to the case when all polynomials in G_r are 0 and $b \neq 0$: $(G_r, \{b\}, G_m)$. Notice that $(G_r, \{b\})$ is consistent, which is detected using the *ZeroDimCheck*.

(3) The next case to consider is when $b = 0$. The Gröbner basis of $G_r \cup \{b\}$ is $\{a^3, ac^2 - a, b\}$. This is the input E' in the recursive call of PGBMain, with the other input being $N' = \{1\}$ and $F' = G \setminus G_r$. It is easy to see that (E', N') is consistent. The reduced Gröbner basis for $F' \cup E'$ is: $G' = \{x^3 - y^3, cx^2 - y, cy^2 - x, a, b\}$ of which $G'_r = \{a, b\}$. It is easy to check the parametric constraint (E', G'_r) is inconsistent: the check for a being in the radical ideal of E' is confirmed by *Icheck*; b is in the ideal of E' . So no branch is generated from this case.

$G'_m = \text{Noncomparable}(G' \setminus G'_r) = \{cx^2 - y, cy^2 - x\}$ and $h' = \text{lcm}\{\text{lc}_X(cx^2 - y), \text{lc}_X(cy^2 - x)\} = c$. This results in another branch: $(G'_r, \{c\}, G'_m)$.

(4) For the case when $h' = c = 0$, $E'' = \{a, b, c\}$ is the Gröbner basis of $G'_r \cup \{c\}$ and $N'' = \{1\}$, $F'' = \{x^3 - y^3, cx^2 - y, cy^2 - x\}$. The Gröbner basis for $F'' \cup E''$ is $G'' = \{x, y, a, b, c\}$. Then $G''_r = \{a, b, c\}$ and $G''_m = \{x, y\}$. Since $h'' = \text{lcm}\{\text{lc}_X(x), \text{lc}_X(y)\} = 1$, this gives another branch: $(G''_r, \{1\}, G''_m)$. As $h'' = 1$, no other branches are created and the algorithm terminates.

The result is a comprehensive Gröbner system for F :

$$\left\{ \begin{array}{ll} \{1\}, & \text{if } a^6 - b^6 \neq 0 \text{ or } a^3c - b^3 \neq 0 \text{ or } b^3c - a^3 \neq 0 \text{ or } ac^2 - a \neq 0 \text{ or } bc^2 - b \neq 0, \\ \{bx - acy, by - a\}, & \text{if } a^6 - b^6 = a^3c - b^3 = b^3c - a^3 = ac^2 - a = bc^2 - b = 0 \text{ and } b \neq 0, \\ \{cx^2 - y, cy^2 - x\} & \text{if } a = b = 0 \text{ and } c \neq 0, \\ \{x, y\} & \text{if } a = b = c = 0. \end{array} \right.$$

7. IMPLEMENTATION AND COMPARATIVE PERFORMANCE

The proposed algorithm is implemented in the system *Magma* and has been experimented with a number of examples from different application domains including geometry theorem proving and computer vision. Since the algorithm is able to avoid most unnecessary branches and computations, it is efficient and can compute comprehensive Gröbner systems for most problems in a few seconds. In particular, we have been successful in completely solving the

famous P3P problem for pose-estimation from computer vision, which is investigated by Gao et al [5] using the characteristic set method; see the polynomial system below.

We have compared our implementation with the implementations of Suzuki and Sato's algorithm as well as Nabeshima's speed-up version as available in the PGB (ver20090915) package implemented in Asir/Risa system. We have picked examples F3, F5, F6 and F8 from [12] and the examples E4 and E5 from [11]; many other examples can be solved in essentially no time. To get more complex examples, we modified problems from the F5, F6 and F8 in [12] slightly, and they are labeled as S1, S2 and S3.

The polynomials for S1, S2, S3 and P3P are given below: $S1 = \{ax^2y + bx^2 + y^3, ax^2y + bxy + cy^2, ay^3 + bx^2y + cxy\}$, $X = \{x, y\}$, $U = \{a, b, c\}$; $S2 = \{x^4 + abx^3 + bcx^2 + cdx + da, 4x^3 + 3abx^2 + 2bcx + cd\}$, $X = \{x\}$, $U = \{a, b, c, d\}$; $S3 = \{ax^2 + byz + c, cw^2 + by + z, (x - z)^2 + (y - w)^2, 2dxw - 2byz\}$, $X = \{x, y, z, w\}$, $U = \{a, b, c, d\}$; $P3P = \{(1 - a)y^2 - ax^2 - py + arxy + 1, (1 - b)x^2 - by^2 - qx + brxy + 1\}$, $X = \{x, y\}$, $U = \{p, q, r, a, b\}$.

Table 1: Timings

Example	Algorithm	Sys.	Br.	time(sec.)
F3	<i>pgbM</i>	Magma	6	0.016
	Suzuki-Sato	Risa/Asir	31	0.5148
	Nabeshima	Risa/Asir	22	0.8268
F5	<i>pgbM</i>	Magma	8	0.016
	Suzuki-Sato	Risa/Asir	11	0.0156
	Nabeshima	Risa/Asir	54	16.04
F6	<i>pgbM</i>	Magma	8	0.078
	Suzuki-Sato	Risa/Asir	875	35.97
	Nabeshima	Risa/Asir	17	0.078
F8	<i>pgbM</i>	Magma	18	0.140
	Suzuki-Sato	Risa/Asir	—	> 1h
	Nabeshima	Risa/Asir	—	> 1h
E4	<i>pgbM</i>	Magma	9	0.016
	Suzuki-Sato	Risa/Asir	15	0.0468
	Nabeshima	Risa/Asir	24	0.7644
E5	<i>pgbM</i>	Magma	38	0.546
	Suzuki-Sato	Risa/Asir	98	24.09
	Nabeshima	Risa/Asir	102	12.53
S1	<i>pgbM</i>	Magma	29	3.167
	Suzuki-Sato	Risa/Asir	—	> 1h
	Nabeshima	Risa/Asir	—	> 1h
S2	<i>pgbM</i>	Magma	15	1.420
	Suzuki-Sato	Risa/Asir	—	> 1h
	Nabeshima	Risa/Asir	49	5.413
S3	<i>pgbM</i>	Magma	30	3.182
	Suzuki-Sato	Risa/Asir	—	> 1h
	Nabeshima	Risa/Asir	—	> 39m Error
P3P	<i>pgbM</i>	Magma	42	6.256
	Suzuki-Sato	Risa/Asir	—	> 1h
	Nabeshima	Risa/Asir	—	> 28m Error

In the above table, the algorithm *pgbM* is the proposed algorithm; the algorithm *cgs1* stands for the Suzuki-Sato's algorithm, and the algorithm *cgs_con1* stands for the Nabeshima's algorithm from Nabeshima's PGB package [13] were used. All the timings in the table are obtained on Core2 Duo3.0 with 4GB Memory running WinVista64.

As is evident from Table 1, the proposed algorithm gen-

erates fewer branches. This is why our algorithm has better performance than the others.

An efficient check for the consistency of parametric constraints is critical for the performance of the proposed algorithm. The role of various checks discussed in Section 5 has been investigated in detail. This is reported in Table 2 below, where *Tri*, *0-dim*, *C*, *I*, and *Gen* stand, respectively, for the *trivial check*, *Zero-DimCheck*, the *CCheck*, *ICheck*, and the *general method*.

Table 2: Info about various consistence checks

Exp		Tri.	0-dim	pos-dim		Gen.	Total
				C.	I.		
F3	Num	10	2	3	0	0	15
	≈ %	67%	13%	20%	0%	0%	
F5	Num	22	0	10	0	0	32
	≈ %	69%	0%	31%	0%	0%	
F6	Num	22	0	7	8	1	38
	≈ %	58%	0%	18%	21%	3%	
F8	Num	47	0	29	0	0	76
	≈ %	62%	0%	38%	0%	0%	
E4	Num	10	7	3	0	0	20
	≈ %	50%	35%	15%	0%	0%	
E5	Num	67	10	55	0	6	138
	≈ %	49%	7%	40%	0%	4%	
S1	Num	115	21	36	0	11	183
	≈ %	63%	11%	20%	0%	6%	
S2	Num	36	0	27	6	0	69
	≈ %	52%	0%	39%	9%	0%	
S3	Num	110	9	45	1	0	165
	≈ %	67%	5%	27%	1%	0%	
P3P	Num	144	4	63	3	13	227
	≈ %	63%	2%	28%	1%	6%	

About 61% of the consistency check is settled by the trivial check that a polynomial is in the ideal; about the remaining 36% of the consistency check is resolved by the *Zero-DimCheck*, *CCheck* and *ICheck*. The general method for checking consistency using Rabinovitch's trick of introducing a new variable is rarely used (almost 3%). We believe that this is one of the main reasons why our proposed algorithm has a vastly improved performance over Nabeshima's speed-up algorithm which relies on using the general check for the consistency of the parametric constraints.

8. CONCLUDING REMARKS

A new algorithm for computing a comprehensive Gröbner system has been proposed using ideas from Kalkbrener, Weispfenning, Suzuki and Sato. Preliminary experiments suggest that the algorithm is far superior in practice in comparison to Suzuki and Sato's algorithm as well as Nabeshima's speed-up version vis a vis the number of branches generated as well as execution speed. Particularly, we are able to do examples such as the famous P3P problem from computer vision, which have been found extremely difficult to solve using most symbolic computation algorithms.

We believe that the proposed algorithm can be further improved. We are exploring conditions under which the radical membership ideal check is unwarranted and additional ideas to make this check more efficient whenever it is needed. We also plan to compare our implementation with other implementations of comprehensive Gröbner system algorithms.

9. REFERENCES

- [1] Becker, T. and Weispfenning, V. (1993). Gröbner Bases, A Computational Approach to Commutative Algebra. Springer-Verlag. ISBN 0-387-97971-9.
- [2] Chen, X.F., Li, P., Lin, L., Wang, D.K. (2005) Proving geometric theorems by partitioned-parametric Gröbner bases. In: Hong, H., Wang, D. (eds.) ADG 2004. LNAI, vol. 3763, 34-44. Springer.
- [3] Cox, D., Little, J., O'Shea, D. (2004). Using Algebraic Geometry. New York, Springer. 2nd edition. ISBN 0-387-20706-6.
- [4] Donald, B., Kapur, D., and Mundy, J.L. (eds.) (1992). Symbolic and Numerical Computation for Artificial Intelligence. Academic Press.
- [5] Gao, X.S., Hou, X., Tang, J. and Chen, H. (2003). Complete Solution Classification for the Perspective-Three-Point Problem, IEEE Tran. on PAMI, 930-943, 25(8).
- [6] Kalkbrener, K. (1997). On the stability of Gröbner bases under specialization, J. Symb. Comp. 24, 1, 51-58.
- [7] Kapur, D. (1995). An approach to solving systems of parametric polynomial equations. In: Saraswat, Van Hentenryck (eds.) Principles and Practice of Constraint Programming, MIT Press, Cambridge.
- [8] Kapur, D. (2006). A Quantifier Elimination based Heuristic for Automatically Generating Inductive Assertions for Programs, J. of Systems Science and Complexity, Vol. 19, No. 3, 307-330.
- [9] Manubens, M. and Montes, A. (2006). Improving DISPG Algorithm Using the Discriminant Ideal, J. Symb. Comp., 41, 1245-1263.
- [10] Montes, A. (2002). A new algorithm for discussing Gröbner basis with parameters, J. Symb. Comp. 33, 1-2, 183-208.
- [11] Montes, A., Recio, T. (2007). Automatic discovery of geometry theorems using minimal canonical comprehensive Gröbner systems. ADG 2006, LNAI 4869, Springer, 113-138.
- [12] Nabeshima, K. (2007) A Speed-Up of the Algorithm for Computing Comprehensive Gröbner Systems. In Brown, C., editor, ISSAC2007, 299-306.
- [13] Nabeshima, K. (2007) PGB: A Package for Computing Parametric Gröbner Bases and Related Objects. Conference posters of ISSAC 2007, 104-105.
- [14] Suzuki, A. and Sato, Y. (2002). An alternative approach to Comprehensive Gröbner bases. In Mora, T., editor, ISSAC2002, 255-261.
- [15] Suzuki, A. and Sato, Y. (2004) Comprehensive Gröbner Bases via ACG. In Tran, Q-N., editor, ACA2004, 65-73.
- [16] Suzuki, A. and Sato, Y. (2006) A Simple Algorithm to compute Comprehensive Gröbner Bases using Gröbner bases. In ISSAC2006, 326-331.
- [17] Wang, D.K. and Sun, Y. (2009) An Efficient Algorithm for Factoring Polynomials over Algebraic Extension Field. arXiv:0907.2300v1.
- [18] Weispfenning, V. (1992). Comprehensive Gröbner bases, J. Symb. Comp. 14, 1-29.
- [19] Weispfenning, V. (2003). Canonical Comprehensive Gröbner bases, J. Symb. Comp. 36, 669-683.

Finding All Bessel Type Solutions for Linear Differential Equations with Rational Function Coefficients

Mark van Hoeij* & Quan Yuan
Florida State University, Tallahassee, FL 32306-3027, USA
hoeij@math.fsu.edu & qyuan@math.fsu.edu

ABSTRACT

A linear differential equation with rational function coefficients has a Bessel type solution when it is solvable in terms of $B_\nu(f)$, $B_{\nu+1}(f)$. For second order equations, with rational function coefficients, f must be a rational function or the square root of a rational function. An algorithm was given by Debeerst, van Hoeij, and Koepf, that can compute Bessel type solutions if and only if f is a rational function. In this paper we extend this work to the square root case, resulting in a complete algorithm to find all Bessel type solutions.

Categories and Subject Descriptors

G.4 [Mathematical Software]: Algorithm design and analysis; I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—Algebraic algorithms

General Terms

Algorithms

1. INTRODUCTION

Let $a_0, a_1, a_2 \in \mathbb{C}(x)$ and let $L = a_2\partial^2 + a_1\partial + a_0$ be a differential operator of order two. The corresponding differential equation is $L(y) = 0$, i.e. $a_2y'' + a_1y' + a_0y = 0$. Let $B_\nu(x)$ denote one of the Bessel functions (one of Bessel I , J , K , or Y functions). The question studied in [6, 7] is the following: Given L , decide if there exists a rational function $f \in \mathbb{C}(x)$ such that L has a solution y that can be expressed¹ in terms of $B_\nu(f)$. If so, then find f , ν , and the corresponding solutions of L . The same problem was also solved for Kummer/Whittaker functions, see [6]. This means that for second order L , with rational function coefficients, there is an almost-complete algorithm in [6] to decide if $L(y) = 0$ is solvable in terms of ${}_0F_1$ or ${}_1F_1$ functions, and if so, to find the solutions.

*Supported by NSF grant 0728853

¹using sums, products, differentiation, and exponential integrals (see Definition 2)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

The reason this almost-complete algorithm is not complete is the following: If $B_\nu(f)$ satisfies a second order linear differential equation with rational function coefficients, then either: $f \in \mathbb{C}(x)$, or (square root case): $f \notin \mathbb{C}(x)$ but $f^2 \in \mathbb{C}(x)$.

However, only the $f \in \mathbb{C}(x)$ case was handled in [6, 7], the square-root case was listed in the conclusion of [7] as a task for future work. This meant that [6, 7] is not yet a complete solver for ${}_0F_1$ and ${}_1F_1$ type solutions.

In this paper, we treat the square-root case for Bessel functions. The combination of this paper with the treatment of Kummer/Whittaker functions in [6] is then a complete algorithm to find ${}_0F_1$ and ${}_1F_1$ type solutions whenever they exist².

The reason why the square-root case was not yet treated in [7] will be explained in the next two paragraphs. If f is a rational function $f = A/B$, then from the generalized exponents at the irregular singularities, we can compute B , as well as $\deg(A)$ linear equations for the coefficients of A , see [7], or see [6] which contains more details and examples. Since a polynomial A of degree $\deg(A)$ has $\deg(A) + 1$ coefficients, this meant that only one more equation was needed to reconstruct A , and in each of the various cases in [6, 7] there was a way to compute such an equation.

In the square-root case, we can not write f as a quotient of polynomials, but we can write $f^2 = A/B$. The same method as in [6, 7] will still produce B , and linear equations for the coefficients of A . The number of linear equations for the coefficients of A is still the same as it was in the $f \in \mathbb{C}(x)$ case. Unfortunately, by squaring f to make it a rational function, we doubled the degree of A , but we do not get more linear equations, which means that in the square-root case the number of linear equations is only $\frac{1}{2}\deg(A)$ (plus an additional ≥ 0 equations coming from regular singularities). So in the worst case, the number of equations is only half of the degree of A . This is why the square-root case was not solved in [7] but only mentioned as a future task.

Our approach is as follows: One can rewrite $A = CA_1A_2^d$ where A_1 can be computed from the regular singularities, but A_2 can not. The problem is that while the degree of A_2 is only $\frac{1}{d}$ times the degree of A/A_1 , the linear equations on the coefficients of A translate into polynomial equations (with degree d) for the coefficients of A_2 . Solving systems of polynomial equations can take too much CPU time. However, we discovered that with some modifications, one

²Other ${}_0F_1$ and ${}_1F_1$ type functions can be rewritten in terms of Bessel, or Kummer/Whittaker functions. For instance, Airy type functions form a subclass of Bessel type functions (provided that the square-root case is treated!)

can actually obtain linear equations for the coefficients of A_2 . This means that we only need to solve linear systems. The result is an efficient algorithm that can handle complicated inputs. An implementation is available online at <http://www.math.fsu.edu/~qyuan>.

2. PRELIMINARIES

2.1 Differential Operators

We let $K[\partial]$ be the ring of differential operators with coefficients in a differential field K . Let C_K be constants, $\overline{C_K}$ be an algebraic closure of C_K . Usually, we have $K = \overline{C_K}(x)$ and C_K is a finite extension of $\mathbb{Q}$.

We call $p \in \overline{C_K} \cup \{\infty\}$ a *singularity* of the differential operator $L \in K[\partial]$, if $p = \infty$ or p is a zero of the leading coefficient of L or p is a pole of a coefficient of L . If p is not a singularity, p is *regular*.

We say y is a solution of L , if $L(y) = 0$. The vector space of solutions is denoted as $V(L)$. If p is regular, we can express all solutions around p as power series $\sum_{i=0}^{\infty} b_i t_p^i$ where t_p denotes the local parameter which is $t_p = \frac{1}{x}$ if $p = \infty$ and $t_p = x - p$, otherwise.

2.2 Formal Solutions and Generalized Exponents

DEFINITION 1. We say $e \in \mathbb{C}[t_p^{-\frac{1}{m}}]$ is a *generalized exponent* of L at p if L has a solution $\exp\left(\int \frac{e}{t_p} dt_p\right) S$, $S \in R_m$, and $S \notin t_p^{\frac{1}{m}} R_m$, where $m \in \mathbb{Z}$ and $R_m = \mathbb{C}[[t_p^{\frac{1}{m}}]][\log(t_p)]$

If $e \in \mathbb{C}$ we just get a solution $x^e S$, in this case e is called an *exponent*. If the solution involves a logarithm, we call it a *logarithmic solution*. If $m = 1$, then e is *unramified*, otherwise it is *ramified*.

REMARK 1. Since we only consider second order differential operators, m in the definition can be only 1 or 2.

If the order of L is n , then at every point p , counting with multiplicity, there are n generalized exponents $e_1, e_2, \dots, e_n$, and the solutions $\exp\left(\int \frac{e_i}{t_p} dt_p\right) S_i$, $i = 1, \dots, n$ are a basis of solution space $V(L)$. If p is regular, then the generalized exponents of L at p are $0, 1, \dots, n-1$. One can compute generalized exponents with the Maple command `DEtools[gen_exp]`.

2.3 Bessel Functions

Bessel functions are the solutions of the operators $L_{B1} = x^2 \partial^2 + x \partial + (x^2 - \nu^2)$ and $L_{B2} = x^2 \partial^2 + x \partial - (x^2 + \nu^2)$. The two linearly independent solutions $J_\nu(x)$ and $Y_\nu(x)$ of L_{B1} are called Bessel functions of first and second kind, respectively. Similarly the solutions $I_\nu(x)$ and $K_\nu(x)$ of L_{B2} are called the modified Bessel functions of first and second kind. Let $B_\nu(x)$ refer to one of the Bessel functions.

When ν is half integer, L_{B1} and L_{B2} are reducible. One can get the solutions by factoring the operators. We will exclude this case from this paper.

The change of variables $x \rightarrow ix$ sends $V(L_{B1})$ to $V(L_{B2})$ and vice versa. Since our algorithm will deal with change of variables, as well as two other transformations (see Section 2.4), we only need one of L_{B1} , L_{B2} . We choose L_{B2} and denote $L_B := L_{B2}$.

L_B has only two singularities, 0 and ∞ . The generalized exponents are $\pm\nu$ at 0 and $\pm t_\infty^{-1} + \frac{1}{2}$ at ∞ .

After a change of variables $y(x) \rightarrow y(\sqrt{x})$, we get a new operator $L_B^\vee = x^2 \partial^2 + x \partial - \frac{1}{4}(x + \nu^2)$, which is still in $\mathbb{Q}(x)[\partial]$. Let $\text{CV}(L, f)$ denote the operator obtained from L by change of variables $x \mapsto f$. For any differential field extension K of $\mathbb{Q}(x)$, if $\nu^2 \in C_K$, and if $f^2 \in K$, then $\text{CV}(L_B, f) \in K[\partial]$ since this operator can be written as $\text{CV}(L_B^\vee, f^2)$. The converse is also true:

LEMMA 1. Let K be a differential field extension of $\mathbb{Q}(x)$, let f, ν be elements of a differential field extension of K , and ν be constant. Then

$$\text{CV}(L_B, f) \in K[\partial] \iff f^2 \in K \text{ and } \nu^2 \in C_K.$$

PROOF. It remains to prove $\implies$. Let ν be a constant, $\text{monic}(L)$ be the differential operator divided by the leading coefficient of L , and

$$M := \text{monic}(\text{CV}(L_B, f)) = \partial^2 + a_1 \partial + a_0$$

We have to prove

$$a_0, a_1 \in K \implies f^2, \nu^2 \in K$$

and so we assume $a_0, a_1 \in K$. Let $g = f^2$. By computing $M = \text{monic}(\text{CV}(L_B^\vee, g))$ we find

$$a_1 = -\text{ld}(\text{ld}(g)), \quad a_0 = \frac{-1}{4}(g + \nu^2)\text{ld}(g)^2$$

where ld denotes the logarithmic derivative, $\text{ld}(a) = a'/a$. Let

$$a_2 := \text{ld}(\text{ld}(a_0) + 2a_1) + \text{ld}(a_0) + 3a_1,$$

$$a_3 := -4a_0/a_2^2, \quad a_4 := a_3(2a_1 + \text{ld}(a_0))$$

which are in K since $a_0, a_1 \in K$. Direct substitution shows that $a_2 = \text{ld}(g)$, $a_3 = g + \nu^2$, and $a_4 = g'$. Hence $g = a_4/a_2 \in K$ and $\nu^2 = a_3 - g \in K$. $\square$

2.4 Transformations

DEFINITION 2. A transformation between two differential operators L_1 and L_2 is an onto map from solution space $V(L_1)$ to $V(L_2)$. For an order 2 operator $L_1 \in K[\partial]$, there are three types of transformations for which the resulting L_2 is again in $K[\partial]$ with order 2. They are (notation as in [6, 7]):

- (i) change of variables: $y(x) \rightarrow y(f(x))$, $f(x) \in K \setminus C_K$.
- (ii) exp-product: $y \rightarrow \exp(\int r dx) \cdot y$, $r \in K$.
- (iii) gauge transformation: $y \rightarrow r_0 y + r_1 y'$, $r_0, r_1 \in K$.

We denote them by $\longrightarrow_C$, $\longrightarrow_E$, $\longrightarrow_G$ respectively.

We can switch the order of $\longrightarrow_E$ and $\longrightarrow_G$ [6]. So we will denote $L_1 \longrightarrow_{EG} L_2$ if some combination of (ii) and (iii) sends L_1 to L_2 . Likewise we denote $L_1 \longrightarrow_{CEG} L_2$ if some combination of (i), (ii), (iii) sends L_1 to L_2 .

REMARK 2. The relation $\longrightarrow_{EG}$ is an equivalence relation. But $\longrightarrow_C$ is not.

DEFINITION 3. We say $L_1 \in K[\partial]$ is *projectively equivalent* to L_2 if and only if $L_1 \longrightarrow_{EG} L_2$.

LEMMA 2. (Lemma 3 in [7]) If $L_1 \longrightarrow_{CEG} L_2$, then there exist an operator $M \in K[\partial]$ such that $L_1 \longrightarrow_C M \longrightarrow_{EG} L_2$.

We can apply Lemma 2 to $L_1 = L_B^\vee$ and L_2 which is the operator L we want to solve. If M is known (i.e if the change of variables $x \rightarrow f$ is known), then the map from $V(M)$ to $V(L)$ can be computed with existing algorithms [1], [6]. That means $L_B^\vee \rightarrow_{CEG} L$ can be computed if we can find the change of variables.

The goal of this paper is to solve differential equations in terms of Bessel functions. This means: if $L_B^\vee \rightarrow_{CEG} L$, then solve L .

Main problem: Let C_K be a field, $C_K \subseteq \mathbb{C}$, and let $K = C_K(x)$. Let $L \in K[\partial]$ be irreducible and of order 2. The question we will solve in this paper is the following: Does there exist an operator $M \in K[\partial]$ such that

1. L is projectively equivalent to M , and
2. $L_B^\vee \xrightarrow{g} M$ for some $g \in K$ and some constant ν .

If so, find g, ν and solve L .

Note that $L_B^\vee \xrightarrow{g} M$ is the same as $L_B \xrightarrow{f} M$, where $f^2 = g \in K$. The reason we use the second form is because we can then use the same notation as in [6] [7].

3. THE CHANGE OF VARIABLES

3.1 The Exponent Difference

To recover f in the transformation $\rightarrow_C$, we need some information about f and this information should be invariant under projective equivalence. Since the order of L is 2, we have two exponents e_1, e_2 at a point p . We consider the exponent difference $\Delta(L, p) = \pm(e_1 - e_2)$. One can verify that Δ modulo $\frac{1}{m}\mathbb{Z}$ is invariant under projective equivalence [6] (Here m is as in Section 2.2). We use a $\pm$ sign because we don't know the order of the generalized exponents. We also define:

DEFINITION 4. A singularity p of $L \in K[\partial]$ is called:

- (i) *apparent singularity* if and only if $\Delta(L, p) \in \mathbb{Z}$ and L is not logarithmic at p . This is equivalent to saying that L has a basis of solutions y_1, y_2 for which y_1/y_2 is analytic at p .
- (ii) *regular singularity* if and only if $\Delta(L, p) \in \mathbb{C} \setminus \mathbb{Z}$ or L is logarithmic at p .
- (iii) *irregular singularity* if and only if $\Delta(L, p) \in \mathbb{C}[t_p^{-\frac{1}{2}}] \setminus \mathbb{C}$

we also denote the set of regular singularities and irregular singularities by S_{reg} and S_{irr} .

Note: Apparent singularities are neither in S_{reg} nor S_{irr} .

The main work in this paper is to construct a finite set of candidates for (f, ν) from the $\Delta(L, p)$. Let $g = f^2 \in K$. If g has a root resp. pole at p of order $k \in \mathbb{N}$, then we say that f has a root resp. pole at p of order $m_p := \frac{k}{2}$.

THEOREM 1. Let $K = C_K(x)$, and $L_B \xrightarrow{f} M \rightarrow_{EG} L$, where $f^2 \in K$. Note: L is the input to our algorithm, and f and M are to be computed.

- (i) if p is a zero of f with multiplicity $m_p \in \frac{1}{2}\mathbb{Z}^+$, then p is an apparent singularity or $p \in S_{reg}$, and $\Delta(M, p) = 2m_p\nu$.

- (ii) p is a pole of f with pole order $m_p \in \frac{1}{2}\mathbb{Z}^+$ such that $f = \sum_{i=-m_p}^{\infty} f_i t_p^i$, if and only if $p \in S_{irr}$ and $\Delta(M, p) = 2 \sum_{i < 0} i f_i t_p^i$.

If $p \in S_{reg}$, then $\Delta(L, p) \equiv \Delta(M, p) \pmod{\mathbb{Z}}$ which means that we can compute $2m_p\nu \pmod{\mathbb{Z}}$.

If $p \in S_{irr}$, then $\Delta(L, p) \equiv \Delta(M, p) \pmod{\frac{1}{m}\mathbb{Z}}$. Then $\sum_{i < 0} f_i t_p^i$ can be computed from $\Delta(L, p)$ by dividing coefficients by $2i$ (the congruence only affects the t_p^0 -term of Δ , but that term is not used when $p \in S_{irr}$).

PROOF. We can use the same proof in [6]. $\square$

DEFINITION 5. Let $f = \sum_{i=N}^{\infty} a_i x^i$, $N \in \mathbb{Z}$, $a_N \neq 0$. We say that we have a k -term truncated power series for f when the coefficient of $x^N, \dots, x^{N+k-1}$ are known.

REMARK 3. If a k -term truncated series for f is known, then we can compute a k -term truncated series for f^2 .

According to Theorem 1 (ii), from $\Delta(M, p)$, we can get a $[m_p]$ -term truncated series of f at p . In [7], f was assumed to be in K , in which case the truncated series is exactly the polar part of f at p . But in this paper, we have to compute $g = f^2 \in K$. Theorem 1 (ii) gives us the polar part of f , i.e. a truncated series for f . We square it to obtain a truncated series of g . But this truncated series for g has $[m_p]$ terms (the same number of terms as the one for f , see Remark 3). So it is only half (rounded up) of the polar part of g . For instance, if f has a pole of order 3 at $x = 0$, from $\Delta(L, p)$ we can obtain a truncated series $\sum_{i=-3}^{-1} a_i x^i$ of f at 0. Squaring this series, we can get the coefficients of x^{-6}, x^{-5}, x^{-4} of g , but not more. So we have:

COROLLARY 1. If $L_B \xrightarrow{f} M \rightarrow_{EG} L$ and $g = f^2$ then we have:

- (i) if $p \in S_{reg}$ then p is a zero of g .
- (ii) $p \in S_{irr}$ if and only if p is a pole of g . We can also get a $[m_p]$ -term truncated series of g from $\Delta(L, p)$, where m_p is the pole order of f .

3.2 The Parameter ν

The exponent difference is also associated with the Bessel parameter ν . As in [6], we have:

THEOREM 2. If $L_B^\vee \rightarrow_{CEG} L$, then

- (i) if $S_{reg} = \emptyset$ then $\nu \in \mathbb{Q} \setminus \mathbb{Z}$.

The following hold for any $p \in S_{reg}$:

- (ii) L logarithmic at p if and only if $\nu \in \mathbb{Z}$.
- (iii) if L is not logarithmic at p and $\Delta(L, p) \in \mathbb{Q}$ then $\nu \in \mathbb{Q} \setminus \mathbb{Z}$.
- (iv) $\Delta(L, p) \in C_K \setminus \mathbb{Q}$ if and only if $\nu \in C_K \setminus \mathbb{Q}$.
- (v) $\Delta(L, p) \notin C_K$ if and only if $\nu \notin C_K$.

We will divide our algorithm into different cases by different situations in Theorem 2. We call (ii) *logarithmic case*. (i) and (iii) *rational case*, and (iv) and (v) *irrational case*. We also have *easy case* which will be defined later. For the logarithmic case, we have

REMARK 4. If any $p \in S_{reg}$ is logarithmic then by Theorem 2 (ii), $\nu \in \mathbb{Z}$, then by again Theorem 2 (ii), we have every $p \in S_{reg}$ must be logarithmic. If not, then L has no Bessel type solutions. Also by the fact $\mathbb{C}(x)B_\nu(x) + \mathbb{C}(x)B'_\nu(x)$ is invariant under $\nu \rightarrow \nu + 1$ and $\nu \rightarrow 1 - \nu$, for the logarithmic case, we can let $\nu = 0$.

For the rational case:

REMARK 5. Since $\mathbb{C}(x)B_\nu(x) + \mathbb{C}(x)B'_\nu(x)$ is invariant under $\nu \rightarrow \nu + 1$ and $\nu \rightarrow 1 - \nu$, if $\nu \in \mathbb{Q}$ then we can just focus on $\nu \in [0, \frac{1}{2}]$. If $\nu = \frac{1}{2}$ the operator will be reducible, it is easy to solve the operator by factoring. So we just consider $\nu \in [0, \frac{1}{2})$.

We will give a method to find a finite list of candidates for ν and f later. If we fix f , then we have:

LEMMA 3. Let Z be the set of all zeroes of f , for $p \in Z$ let m_p be the multiplicity at p .

(i) If $\Delta(L, p) \in C_K$, then let

$$\mathcal{N}'_p := \left\{ \frac{\Delta(L, p) + i}{2m_p} \mid 0 \leq i \leq 2m_p - 1, i \in \mathbb{Z} \right\}$$

We can make the rational part of each element in $\mathcal{N}'_p$ belong to $[0, \frac{1}{2}]$. Let the new set be $\mathcal{N}_p$. Then $\nu \in \mathcal{N} := \bigcup_{p \in S_{reg}} \mathcal{N}_p$.

(ii) If $\Delta(L, p) \notin C_K$, we can write $\Delta(L, p)$ as $a_1\sqrt{k} + a_2$ where $k \in C_K$ and $a_1, a_2 \in C_K$. Then $\nu = \frac{a_1\sqrt{k}}{2m_p}$ (if for different p , we get different ν then there are no Bessel type solutions.)

PROOF. The lemma follows from the fact that we know the number $\Delta(M, p) = 2m_p\nu \bmod \mathbb{Z}$, the fact that $\nu^2 \in C_K$, and the fact that $\mathbb{C}(x)B_\nu(x) + \mathbb{C}(x)B'_\nu(x)$ is invariant under $\nu \rightarrow \nu + 1$ and $\nu \rightarrow 1 - \nu$. $\square$

3.3 Easy, Logarithmic and Irrational Cases

To retrieve f , we need enough linear equations. We assume $L_B \xrightarrow{f} C M \xrightarrow{EG} L$. We want to get information about f from L . Since f might not be in K , but $g = f^2$ is in K , we can assume $g = \frac{A}{B}$, $A, B \in C_K[x]$, B is monic and $\gcd(A, B) = 1$. We want to get information about A, B from L . Since Maple can compute generalized exponents of L , we can compute the exponent difference at singularity p . Then by Corollary 1 we can get the set S_{reg} , which give us some zeroes of g , and S_{irr} , which give the truncated series at each $p \in S_{irr}$. The following two lemmas are true for all cases:

LEMMA 4. We can retrieve B from S_{irr} .

PROOF. According to Theorem 1 (ii), if $p \in S_{irr}$ then p is a pole of f . Let $m_p \in \frac{1}{2}\mathbb{Z}^+$ be the pole order of $\Delta(M, p)$. g has a pole order $2m_p$. The Theorem implies $B = \prod_{p \in S_{irr} \setminus \{\infty\}} (x - p)^{2m_p}$. $\square$

LEMMA 5. Let

$$d_A = \begin{cases} \deg(B) + 2m_\infty & \text{if } \infty \in S_{irr} \\ \deg(B) & \text{otherwise} \end{cases}$$

- (i) If $\infty \in S_{reg}$ then $\deg(A) < d_A$;
- (ii) if $\infty \in S_{irr}$ then $\deg(A) = d_A$;
- (iii) otherwise $\deg(A) \leq d_A$.

In all cases, we can write $A = \sum_{i=0}^{d_A} a_i x^i$, so we have $d_A + 1$ unknowns.

PROOF. According to Corollary 1 (i), if $\infty \in S_{reg}$ then we have $\deg(A) < \deg(B)$. If $\infty \in S_{irr}$ with pole order m_∞ , then $\deg(A) = \deg(B) + 2m_\infty$ (see Corollary 1 (ii)). If $\infty \notin S_{irr}$ then f does not have a pole at ∞ , so that $\deg(A) \leq \deg(B)$. $\square$

LEMMA 6. Assume $p \in C_K$, if $p \in S_{reg}$, we will get one linear equation for the coefficients of A . If $p \in S_{irr}$ with m_p as pole order of $\Delta(L, p)$, we will get $\lceil m_p \rceil$ linear equations.

PROOF. According to Corollary 1 (i), if $p \in S_{reg}$, p is a zero of A . Then we will get a linear equation of $\{a_i\}_{i=0, \dots, d_A}$ by setting $\text{rem}(A, x - p) = 0$.

In addition, for each $p \in S_{irr}$ with pole order m_p , by Corollary 1 (ii) we will have a $\lceil m_p \rceil$ -term truncated series of g at p . Then we can get the truncated series of $A = gB$. On the other hand, we can rewrite $A = \sum_{i=0}^{d_A} a_i x^i$ as a truncated series at p (by Taylor or Laurent series). Since the terms in a Taylor series or Laurent series depend linearly on the coefficients of A , by comparing the coefficients, each term will give a linear equation of a_i . $\square$

EXAMPLE 1.

$$L = \partial^2 + \frac{10x^3 - 21x^2 + 12x - 4}{x(x-1)(x-2)(5x-2)} \partial - \frac{1}{36} \frac{28x^4 - 89x^3 + 105x^2 - 59x + 16}{x^2(x-1)^2(x-2)^6} \partial$$

and $K = \mathbb{Q}(x)$.³ Then we can compute $S_{reg} = \{0\}$, $S_{irr} = \{2\}$ and the truncated series of g at $x = 2$ is $6t_2^{-4} + 21t_2^{-3} + O(t_2^{-2})$, so $B = (x - 2)^4$ and $d_A = 4$. We assume $A = \sum_{i=0}^4 a_i x^i$. Then $\text{rem}(A, x) = a_0 = 0$ give us one linear equation. And since we can rewrite $\frac{A}{B} = (a_0 + 2a_1 + 4a_2 + 8a_3 + 16a_4)t_2^{-4} + (a_1 + 4a_2 + 12a_3 + 32a_4)t_2^{-3} + O(t_2^{-2})$. By comparing the coefficient of two truncated series, we can get 2 linear equations $a_0 + 2a_1 + 4a_2 + 8a_3 + 16a_4 = 6$ and $a_1 + 4a_2 + 12a_3 + 32a_4 = 21$. For this example, we have 5 unknowns and we only have 3 linear equations. But we can still solve it (see Example 3).

For $p \in \overline{C_K}$, $p \notin C_K$, we have:

LEMMA 7. If $p \notin C_K$, let $l(x)$ be the minimal polynomial of p over C_K . If $p \in S_{reg}$, we will have $\deg(l)$ linear equations. If $p \in S_{irr}$, we will have $\deg(l) \cdot \lceil m_p \rceil$ linear equations.

PROOF. If $p \in S_{reg}$, then p is a zero of g . Then all the conjugates in $\overline{C_K}$ of p are zeroes of g . There are $\deg(l)$ conjugated zeroes and by setting $\text{rem}(A, l(x)) = 0$, we will get $\deg(l)$ linear equations with coefficient in C_K .

If $p \in S_{irr}$ with m_p as pole order of $\Delta(L, p)$, we can first consider it in the field $C_K(p)$. Then according to lemma 6 we will get $\lceil m_p \rceil$ linear equations with coefficients in $C_K(p)$. Let $c + \sum_{i=0}^n c_i a_i = 0$ be such an equation, where $\{a_i\}$ are unknowns and $\{c_i\}$ are coefficients in $C_K(p)$. We can rewrite the equation as $\sum_{i=0}^{\deg(l)-1} e_i p^i = 0$ where $\{e_i\}$ are linear functions with coefficients in C_K . Now p is algebraic over C_K of degree $\deg(l)$, so $1, p, \dots, p^{\deg(l)-1}$ are linearly independent over C_K . Hence the e_i are 0; we get $\deg(l)$ linear equations over C_K . We can do this for each of the $\lceil m_p \rceil$ linear equations. Then we get $\deg(l) \cdot \lceil m_p \rceil$ equations. $\square$

EXAMPLE 2. Suppose $\sqrt{2} \in S_{reg}$. If $\sqrt{2} \in C_K$, we get one equation by $\text{rem}(A, x - \sqrt{2}) = 0$. If $\sqrt{2} \notin C_K$, we get two equations by $\text{rem}(A, x^2 - 2) = 0$.

Suppose $\sqrt{2} \in S_{irr}$, and that one of the $\lceil m_{\sqrt{2}} \rceil$ linear equations is $3 + (1 - \sqrt{2})a_1 + (1 + \sqrt{2})a_2 = 0$. If $\sqrt{2} \notin C_K$, we can rewrite that equation as $(3 + a_1 + a_2) + (a_2 - a_1)\sqrt{2} = 0$. Then we can get two equations $\{3 + a_1 + a_2 = 0, a_2 - a_1 = 0\}$.

³the data is from examples at www.math.fsu.edu/~qyuan

So far, we get at least $\#S_{reg} + \frac{1}{2}d_A$ linear equations for the coefficients of A . If this number is greater than $\deg(A)$, then we can solve them and find A . We call this case the easy case. This case is very similar to the case in [7].

For the logarithmic case and irrational case, we have:

LEMMA 8. *In the logarithmic and the irrational case, we know all zeroes of A . In the irrational case, we know their multiplicities as well.*

PROOF. By Theorem 1 (i), a change of variables can transfer a regular singularity to an apparent singularity only if $\nu \in \mathbb{Q} \setminus \mathbb{Z}$. So in the logarithmic and irrational cases, S_{reg} contains all zeroes.

In the irrational case, for each $p \in S_{reg}$, let a_p be the coefficient of the irrational part of the exponent difference. Then there exists k , such that $k \sum_{p \in S_{reg}} a_p = d_A$. Then $\frac{a_p}{k}$ will give the multiplicity of p . $\square$

In the irrational case, there is only one unknown coefficient, the leading coefficient of A . But we have $\frac{1}{2}d_A$ linear equations, enough to get A .

In the logarithmic case, we have to do a combinatorial search: try all possible combinations of multiplicities of zeroes of A . After that the only unknown is the leading coefficient of A . We have enough equations to find it.

Once we get f , we can get a list of ν by Lemma 3 and Remark 4.

3.4 Rational Case

The hardest case is the rational case. We will compute (see Lemma 9 and 10) a finite set of possible values for $d = \text{denom}(\nu)$. We notice that $d > 2$ because the case $\nu \in \mathbb{Z}$ has already been treated (logarithmic case) and if $\nu \in \frac{1}{2}\mathbb{Z}$ then L_B is reducible. Let $L_B \xrightarrow{f} C M \xrightarrow{EG} L$, $g = f^2 = \frac{A}{B}$. Let p be a root of A and $\Delta(L, p) \equiv 2m_p \nu \pmod{\mathbb{Z}}$. If $d \mid 2m_p$, change of variables $x \mapsto f$ will send p to an apparent singularity. This is hard because if p is apparent, then $p \notin S_{reg}$, which means that not all roots of A are known (not all roots of A are in S_{reg}). But if a zero p of A becomes an apparent singularity, the multiplicity $2m_p$ ⁴ must be a multiple of d . So we can rewrite $A = CA_1A_2^d$, where $A_1, A_2 \in C_K[x]$ and $C \in C_K$, A_1 is monic and the roots of A_1 are the known roots of A (the elements of S_{reg}).

For $S_{reg} = \emptyset$, we can let $A_1 = 1$ and fix d by the following lemma [7]:

LEMMA 9. *If $S_{reg} = \emptyset$, then $d \mid d_A$.*

For $S_{reg} \neq \emptyset$, we have:

LEMMA 10. *If $S_{reg} \neq \emptyset$, we can find a list of candidate pairs (d, A_1) by solving an equation.*

PROOF. We assume $N = \#S_{reg}$, $S_{reg} = \{p_1, \dots, p_N\}$ and $\Delta(L, p)$ is the exponent difference at p . Let $A_1 = \prod_{i=1}^N (x - p_i)^{m_{p_i}}$, $1 \leq m_{p_i} < d$ and $d_p = \text{denom}(\Delta(L, p))$. For each point $p \in S_{reg}$, $d_p \mid d$. So we have $l \mid d$ where $l := \text{lcm}_{p \in S_{reg}} d_p$. So d can only be a multiple of l , and it must be $\leq d_A$. So there are $\lfloor d_A/l \rfloor$ possibilities for d . Once we fix d , then for each $p \in S_{irr}$ we have $\frac{d}{d_p} \mid m_p$. So solve $(\sum_{i=1}^N m_{p_i}) + \deg(A_2)d = d_A$, $1 \leq m_{p_i} < d$ and $\frac{d}{d_{p_i}} \mid m_{p_i}$. It will give finitely many candidates for A_1 . $\square$

⁴If m_p is multiplicity of f at p , then $2m_p$ is multiplicity of A .

LEMMA 11. *We can choose C , such that the d th root of the coefficient of the initial term of the truncated series of $A/(CA_1)$ at p is in C_K .*

PROOF. If $(C_K \cup \infty) \cap S_{irr} = \emptyset$, then extend C_K so that it contains at least one element of S_{irr} . Choose $\tilde{p} \in (C_K \cup \infty) \cap S_{irr}$. From $\Delta(L, \tilde{p})$, we can compute a truncated series for $f^2 = \frac{CA_1A_2^d}{B}$. From it, we can compute a truncated series for f^2B/A_1 . Let C be the coefficient of the first term of this series, which will finish the proof (note that $f^2B/A_1 = CA_2^d$). $\square$

Now the only unknown part of A is A_2 . We can assume $A_2 = \sum_{i=0}^{\deg(A_2)} b_i x^i$. Since $\deg(A_2) \leq \frac{1}{d}d_A \leq \frac{1}{3}d_A$, we have

LEMMA 12. *For the rational case, we only need $\frac{1}{3}d_A + 1$ equations to recover A .*

We can not get the equations by the same methods as in Lemma 6 and [6, 7]. If we do so, the equations we get for $\{b_i\}$ will not be linear. The solution to this problem is as follows:

THEOREM 3. *In the rational case, for $A = CA_1A_2^d$, and $A_2 = \sum_{i=0}^{\deg(A_2)} b_i x^i$, for each $p \in S_{irr}$ with m_p as pole order of exponent difference, if $p \in C_K$, we will get $\lceil m_p \rceil$ linear equations of $\{b_i\}$.*

PROOF. Since the exponent difference at p will give a $\lceil m_p \rceil$ -term truncated series of $g = \frac{A}{B}$ at $x = p$, we can also write B and CA_1 as a series at p . Then we can get the $\lceil m_p \rceil$ -term truncated series of $A_2^d = \frac{gB}{CA_1}$. We assume the series is $\sum_{m_p < i \leq 2m_p} c_i t_p^{-i}$ where t_p is the local parameter at p . We can rewrite the series as $c_{2m_p} t_p^{-2m_p} S$, where S is a power series with the initial term 1. Let $S_{1/d}$ be a power series with first term 1 such that $S_{1/d}^d = S$. Write $S_{1/d} = 1 + \sum_{i>0} a_i t_p^i$ where $a_1, \dots, a_{\lceil m_p \rceil - 1}$ are computed by Hensel lifting. Let $\mu_d = \{r \mid r \in C_K, r^d = 1\}$. By Lemma 11 there should be a d th root of c_{2m_p} in C_K . Let c be such a root. Then for each $r \in \mu_d$, let $S_r = c t_p^{-2m_p/d} r S_{1/d}$. Then S_r is a truncated series at p whose d th power is the truncated series of $\frac{gB}{CA_1}$ at p . Then we can also rewrite $A_2 = \sum_{i=0}^{\deg(A_2)} b_i x^i$ as a truncated series at p . By comparing the coefficients of S_r and A_2 , we will get $\lceil m_p \rceil$ linear equations. Doing this for every $p \in S_{irr}$ provides enough linear equations to find A . Note that we have to try all combinations of $r \in \mu_d$ at every $p \in S_{irr}$. $\square$

REMARK 6. *If $p \notin C_K$, we can use the results from Lemma 7 to get equations. So we can always obtain $\geq \frac{1}{2}d_A$ linear equations, while $\lfloor \frac{1}{3}d_A \rfloor + 1$ equations are sufficient. So we always get enough linear equations.*

REMARK 7. *If we get a candidate (f, d) , then $\{f\} \times \{\frac{a}{d} \mid \gcd(a, d) = 1, 1 \leq a < \frac{1}{2}d\}$ is a list of candidates for (f, ν) .*

To sum up, for all different cases, we have:

THEOREM 4. *From $\Delta(L, p)$, we can always get a list of candidates for (f, ν) .*

PROOF. We always have at least $\#S_{reg} + \frac{1}{2}d_A$ linear equations for the coefficients of A . But we may have enough equations (easy case), or only need either 1 (logarithmic case and irrational case) or $\frac{1}{3}d_A + 1$ equations (rational case) to get g . By Remark 4, Remark 5 Remark 7 and Lemma 3, we can also get a finite list of ν . $\square$

The theorem means that we can always find the change of variables. After that, we can compute the projective equivalence to complete the algorithm.

EXAMPLE 3. *Continue with Example 1. We know $S_{reg} = \{0\}$, $S_{irr} = \{2\}$ with the truncated series of g is $\Delta = 6t_2^{-4} + 21t_2^{-3} + O(t_2^{-2})$, $B = (x-2)^4$ and $d_A = 4$. Lemma 6 did not provide sufficiently many equations. But for this case the only possible situation is $A = CxA_2^3$, and $A_2 = a_0 + a_1x$. The truncated series at $x = 2$ of CA_2^3 is the series of $\Delta \cdot (x-2)^4/x$ at 2, which is $3 + 9t_2 + O(t_2^2)$. So we can let $C = 3$. Then series of $\frac{gB}{CA_1^3}$ is $S = 1 + 3t_2$. Since $K = \mathbb{Q}(x)$, the only 3rd root of 1 is 1. So the only possible truncated series which is 3rd root of S is $1 + t_2 + O(t_2^2)$. And comparing it with $a_0 + a_1x = a_0 + 2a_1 + a_1t_2$, we get two linear equations $a_0 + 2a_1 = 1$ and $a_1 = 1$. Solve them we get $a_0 = -1, a_1 = 1$. So $g = \frac{3x(x-1)^3}{(x-2)^4}$.*

4. THE ALGORITHM

The input of the algorithm is a differential operator L of order 2. We want to find solutions that can be represented in terms of Bessel functions, if such solutions exist. Otherwise the algorithm outputs $\emptyset$. Algorithm 1 gives the sketch.

Data: an irreducible differential operator L
Result: solutions represented in terms of Bessel functions if they exist
Find all singularities by factoring the leading coefficient of L over C_K ;
foreach Singularity p **do**
 compute the generalized exponents at p , then
 compute the exponent differences and then the truncated series of g
end
Get S_{reg} and S_{irr} according to the generalized exponent differences;
Compute B, d_A (Lemma 4 and 5) and the number of linear equations N ($N \geq \#S_{reg} + \frac{1}{2}d_A$);
if $N > d_A$ **then**
 go to **easy case**
else if L logarithmic at some $p \in S_{reg}$ **then**
 go to **logarithmic case**
else if there is $p \in S_{reg}$ with $\Delta(L, p) \notin \mathbb{Q}$ (i.e. $\nu \notin \mathbb{Q}$) **then**
 go to **irrational case**
else
 go to **rational case**
end
/* It will give us a list of candidates for (f, ν) , where f is the function of the change of variables, and ν is the parameter of Bessel functions */
foreach (f, ν) in list of candidates **do**
 Compute an operator $M_{(f, \nu)}$ such that
 $LB \xrightarrow{f} M_{(f, \nu)}$;
 Use algorithm described in [1] to compute whether $M_{(f, \nu)} \rightarrow_{EG} L$ and compute the transformation;
 if such transformation exists **then**
 Add the solution to Solutions List
 end
end
Output the solutions list;
Algorithm 1: Main Algorithm

Now we will explain the detail how to retrieve f, ν in different cases.

4.1 Easy Case

In this case, we have enough linear equations from Lemma 6 to recover g . After that, we can use Lemma 3 to get ν . See Algorithm 2 for detail.

Data: S_{reg}, S_{irr} with truncated series, B, d_A
Result: potential list of (f, ν)
Find all linear equations described in Lemma 6;
Solve linear equations to find f ;
if there is no solution **then**
 output $\emptyset$
else
 Use Lemma 3 to get a list N of candidate ν 's
end
foreach $\nu \in N$ **do**
 Add (f, ν) to output list
end

Algorithm 2: Easy Case

4.2 Logarithmic Case

By Remark 4, we can let $\nu = 0$. By Lemma 8, we know all the zeroes of g . We do not yet know the leading coefficient and the multiplicity of each zero. So we can try all combinations of possible multiplicities. Algorithm 3 will give the sketch.

Data: S_{reg}, S_{irr} with truncated series, B, d_A
Result: list of (f, ν)
if not every singularity $p \in S_{reg}$ is logarithmic **then**
 output $\emptyset$
else
 Let $\nu = 0, A = a \prod_{p \in S_{reg}} (x - p)^{a_p}$;
 foreach $\{a_p\}$ such that $\sum_{p \in S_{reg}} a_p = d_A$ **do**
 Use linear equations described in Lemma 6 to solve a ;
 if the solution exists **then**
 Add $(\frac{A}{B}, 0)$ to output list
 end
 end
end

Algorithm 3: Logarithmic case

4.3 Irrational Case

In this case, by Lemma 8 we have all the zeroes with multiplicities of g . The only unknown part should be the leading coefficient. But we have at least one linear equations. Algorithm 4 gives the sketch.

Data: S_{reg}, S_{irr} with truncated series, B, d_A
Result: list of (f, ν)
Use Lemma 8 find all zeroes and multiplicities;
Use linear equations given by Lemma 6 to get the leading coefficient;
Use Lemma 3 to get a list of candidates for ν 's;
Add solutions to output list;

Algorithm 4: Irrational case

4.4 Rational Case

This is the most complicated case. Let $d = \text{denom}(\nu)$ and $f^2 = g = \frac{CA_1A_2^d}{B}$. Algorithm 5 gives the sketch.

Data: S_{reg}, S_{irr} with truncated series, B, d_A
Result: list of (f, ν)
if $S_{reg} = \emptyset$ **then**
 Let the list of candidates for d be the set of factors of d_A ;
 Let $A_1 = 1$;
else
 Use Lemma 10 to get a list of candidates for d and A_1
end
foreach candidate (d, A_1) **do**
 Fix C by Lemma 11;
 Use linear equations given by Theorem 3 to compute A_2 ;
 If a solution exists, add $\{f\} \times \{\frac{a}{d} \mid \gcd(a, d) = 1, 1 \leq a < \frac{1}{2}d\}$ to output list
end

Algorithm 5: Rational case

5. EXAMPLES

This section will illustrate the algorithm with a few examples⁵.

EXAMPLE 4. Let $L = \partial^2 + 2 - 10x + 4x^2 - 4x^4$. $K = \mathbb{Q}(x)$
Step 1: We get $S_{reg} = \emptyset$. $S_{irr} = \{\infty\}$ with the truncated series of g at $x = \infty$ is $\frac{4}{9}t_\infty^{-6} - \frac{4}{3}t_\infty^{-4} + O(t_\infty^{-3})$. So $d_A = 6$ and $B = 1$.

Step 2: It is the rational case with $S_{reg} = \emptyset$. So $d \in \{3, 6\}$ and we can write $A = CA_2^d$. If $d = 3$ then $A = CA_2^3$, $A_2 = a_0 + a_1x + a_2x^2$. Since $B = 1$, then the truncated series of gB is the same as g . So we can let $C = \frac{4}{9}$. Then the truncated series of A_2^3 is $t_\infty^{-6} + 3t_\infty^{-4} = t_\infty^{-6}(1 - 3t_\infty^2)$. Since the only 3rd root of 1 in C_K is 1, then the only 3rd root of $1 - 3t_\infty^2$ is $1 - t_\infty^2$. So by comparing coefficients of $t_\infty^{-2}(1 - t_\infty^2)$ and $A_2 = a_0 + a_1t_\infty^{-1} + a_2t_\infty^{-2}$, we can get $A_2 = x^2 - 1$ and then $g = \frac{4}{9}(x^2 - 1)^3$. We can do this process for $d = 6$, in this case, we have no solution. So we have $(\frac{2}{3}\sqrt{(x^2 - 1)^3}, \frac{1}{3})$ as the only possible candidate.

Step 3: We compute $L_B \xrightarrow{f} M$, and then the projective equivalence from M to L . Combining these transformations produces the following solutions of L :

$$\begin{aligned} & C_1 \left(\frac{2(2x^4 + x^3 - 3x^2 + x + 2)}{\sqrt{x^2 - 1}} I_{\frac{1}{3}} \left(\frac{2}{3} \sqrt{(x^2 - 1)^3} \right) \right. \\ & + 2(2x + 1)(x^2 - 1) I_{\frac{4}{3}} \left(\frac{2}{3} \sqrt{(x^2 - 1)^3} \right) \\ & + C_2 \left(\frac{2(2x^4 + x^3 - 3x^2 + x + 2)}{\sqrt{x^2 - 1}} K_{\frac{1}{3}} \left(\frac{2}{3} \sqrt{(x^2 - 1)^3} \right) \right. \\ & \left. - 2(2x + 1)(x^2 - 1) K_{\frac{4}{3}} \left(\frac{2}{3} \sqrt{(x^2 - 1)^3} \right) \right) \end{aligned}$$

⁵More examples are given at <http://www.math.fsu.edu/~qyuan>

EXAMPLE 5. Consider the operator:

$$\begin{aligned} L := \partial^2 - \frac{15x^4 - 30x^3 + x^2 + 8x - 4}{x(x-1)(15x^3 - 10x^2 + 9x - 4)} \partial - \\ \frac{1}{36x^2(15x^3 - 10x^2 + 9x - 4)(x-1)^2} (30375x^{20} - \\ 212625x^{19} + 733050x^{18} - 170595x^{17} + 3034305x^{16} - \\ 435055x^{15} + 5166936x^{14} - 5172228x^{13} + 4401369x^{12} - \\ 3189159x^{11} + 1962738x^{10} - 1016622x^9 + 434943x^8 - \\ 149229x^7 + 38844x^6 - 3933x^5 - 4554x^4 + 3789x^3 - \\ 1612x^2 + 432x - 64). \end{aligned}$$

Step 1: The non-apparent singularities are $\infty, 1, 0$.

Step 2: $S_{reg} = \{1, 0\}$, with the exponent difference $\frac{5}{3}$ and $\frac{4}{3}$ respectively. We also have $S_{irr} = \{\infty\}$ and the truncated series of g at $x = \infty$ is $t_\infty^{-15} - 5t_\infty^{-14} + 13t_\infty^{-13} - 25t_\infty^{-12} + 38t_\infty^{-11} - 46t_\infty^{-10} + 46t_\infty^{-9} - 38t_\infty^{-8} + O(t_\infty^{-7})$. So $B = 1$ and $d_A = 15$.

Step 3: we can easily verify that this is a rational case. Since the exponent difference of at 0 and 1 both have denominator 3, so d is a multiple of 3. If $d = 3$ then $A = Cx^2(x-1)A_2^d$ or $A = Cx(x-1)^2A_2^d$. If $d = 6$, then the multiplicity of both 1 and 0 should be a multiple of $\frac{6}{3} = 2$ then it will contradict with $d_A = 15$. Similarly $A = Cx^3(x-1)^3A_2^9$, $A = Cx^5(x-1)^{10}$ and $A = Cx^{10}(x-1)^5$ are candidates as well. Then we compute each candidate by the method in Theorem 3. Finally, we get $f = \sqrt{x^4(x-1)^5(x^2+1)^3}$ and $\nu = \frac{1}{3}$ is the only remaining candidate.

Step 4: Let $L_B \xrightarrow{f} M$. Now M is already equal to L . So the general solution is:

$$\begin{aligned} & C_1 I_{\frac{1}{3}} (\sqrt{x^4(x-1)^5(x^2+1)^3}) + \\ & C_2 K_{\frac{1}{3}} (\sqrt{x^4(x-1)^5(x^2+1)^3}) \end{aligned}$$

6. CONCLUSION

We developed an algorithm to solve second order differential equations in terms of Bessel functions. We extended the algorithm described in [7] which already solved the problem in the $f \in \mathbb{C}(x)$ case, but not in the square root case. We implemented the algorithm in Maple (available from <http://www.math.fsu.edu/~qyuan>). A future task is to try to develop a similar algorithm to find ${}_2F_1$ type solutions.

7. REFERENCES

- [1] BARKATOU, M. A., AND PFLÜGEL, E. On the Equivalence Problem of Linear Differential Systems and its Application for Factoring Completely Reducible Systems. In *ISSAC 1998*, 268–275.
- [2] BRONSTEIN, M. An improved algorithm for factoring linear ordinary differential operators. In *ISSAC 1994*, 336–340.
- [3] BRONSTEIN, M., AND LAFAILLE, S. Solutions of Linear Ordinary Differential Equations in Terms of Special Functions. In *ISSAC 2002*, 23–28.

- [4] CHAN, L., AND CHEB-TERRAB, E. S. Non-Liouvilian Solutions for Second Order Linear ODEs. In *ISSAC 2004*, 80–86.
- [5] CLUZEAU, T., AND VAN HOEIJ, M. A Modular Algorithm to Compute the Exponential Solutions of a Linear Differential Operator. *J. Symb. Comput.* *38* (2004), 1043–1076.
- [6] DEBEERST, R. *Solving Differential Equations in Terms of Bessel Functions*. Master’s thesis, Universität Kassel, 2007.
- [7] DEBEERST, R., VAN HOEIJ, M., AND KOEPF, W. Solving Differential Equations in Terms of Bessel Functions. In *ISSAC 2008*, 39–46.
- [8] EVERITT, W. N., SMITH, D. J., AND VAN HOEIJ, M. The Fourth-Order Type Linear Ordinary Differential Equations. *arXiv:math/0603516* (2006).
- [9] VAN HOEIJ, M. *Factorization of Linear Differential Operators*. PhD thesis, Universiteit Nijmegen, 1996.
- [10] VAN DER HOEVEN, J. Around the Numeric-Symbolic Computation of Differential Galois Groups. *J. Symb. Comp.* *42* (2007), 236–264.
- [11] VAN DER PUT, M., AND SINGER, M. F. *Galois Theory of Linear Differential Equations*, Springer, Berlin, 2003.
- [12] WILLIS, B. L. An Extensible Differential Equation Solver. *SIGSAM Bulletin* *35* (2001), 3–7.

Simultaneously Row- and Column-Reduced Higher-Order Linear Differential Systems

Moulay A. Barkatou, Carole El Bacha
University of Limoges;
CNRS; XLIM UMR 6172, DMI
87060 Limoges, France
{moulay.barkatou,carole.el-
bacha}@xlim.fr

Eckhard Pflügel
Faculty of CISM
Kingston University
Penrhyn Road
Kingston upon Thames, Surrey KT1 2EE
United Kingdom
E.Pfluegel@kingston.ac.uk

ABSTRACT

In this paper, we investigate the local analysis of systems of linear differential-algebraic equations (DAEs) and second-order linear differential systems. In the first part of the paper, we show how one can transform an input linear DAE into a reduced form that allows for the decoupling of the differential and algebraic components of the system. Classification of singularities of linear DAEs are defined and discussed. In the second part of the paper, we extend this approach to second-order linear differential systems and discuss two applications: the classification of singularities and the computation of regular solutions. The present paper is the first step towards a generalisation of the formal reduction of first-order ODEs to higher-order systems. Our algorithm has been implemented in the computer algebra system Maple as part of the ISOLDE package.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms

General Terms

Algorithms

Keywords

Computer Algebra, Systems of Linear Differential Equations, Reduction Algorithms, Singularities

1. INTRODUCTION

We consider linear differential systems of the form

$$L(y) = \sum_{i=0}^{\ell} A_i(x)y^{(i)}(x) = f(x)$$

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

where x is a complex variable, $A_i(x)$ are $m \times n$ matrices of analytic functions and the right-hand side $f(x)$ is a vector of analytic functions of size m . We are interested in the local analysis of such systems at the point $x = 0$, and therefore can suppose, without loss of generality, that the entries of $A_i(x)$ and $f(x)$ are formal power series. Such systems arise naturally in many applications of multi-body systems, models of electrical circuits, robotic modelling and mechanical systems (see [10, 14, 17] and the references therein). This paper will be mainly focused on algorithms that reduce such a system to an equivalent simpler one. In the first part, we study linear differential-algebraic equations of first-order ($\ell = 1$) of the form

$$L(y) = A(x)y'(x) + B(x)y(x) = f(x). \quad (1)$$

The aim is to develop an algorithm which computes a system equivalent to (1) to which the classical theory of ordinary differential equations (ODEs) is applicable and the existence of solutions can be easily decided. Many works have been developed in this direction, see for example [8, 9, 15]. They consist, roughly speaking, in transforming (1) into a simpler differential system so that there exists a one-to-one correspondence between their respective solution sets. In [9], the authors have developed a numerical reduction algorithm for (1) with continuous matrix coefficients on a real closed interval under the assumption that the leading coefficient A and some other sub-matrices have constant rank on this interval. A symbolic method has been presented in [15, 16], improving that of [9], but the size of the system during execution of this algorithm might increase. An older algorithm has been proposed by Harris et al in [8] which reduces (1) to a sequence of first-order systems of ODEs and algebraic systems of lower sizes and some necessary conditions on the right hand side $f(x)$. We shall review this algorithm in more details in Section 3. Traditionally, linear DAEs have been tackled using the notion of differential index introduced in [7] (a range of alternative index definitions exist as well, see for example [17]). Generally speaking, most authors try to extract the underlying ODE which, by definition, is an explicit ordinary differential system computed by differentiating (1) successively and then using only algebraic manipulations. In this paper however, motivated by the work of Harris et al, we use a different strategy: using the terminology of matrix differential operators we find a sequence of left- and right-transformations in order to compute a new operator that has a decoupled differential and algebraic system. Hence, the first contribution of the present paper is the develop-

ment of a new reduction algorithm that for L given by (1), computes an operator $\tilde{L}$ of the form

$$\tilde{L} = SLT = \tilde{A}\partial + \tilde{B} = \begin{bmatrix} \tilde{A}^{11}\partial + \tilde{B}^{11} & 0 & 0 \\ 0 & \tilde{B}^{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (2)$$

where $\partial = \frac{d}{dx}$, S and T are invertible matrix differential operators (see Definition 2.1) and $\tilde{A}^{11}$ and $\tilde{B}^{22}$ are both invertible. Hence the DAE (1) is reduced to $\tilde{L}(z) = \tilde{f}$, where $y = T(z)$ and $\tilde{f} = S(f)$, which now can be written as two separate problems, possibly of lower size than (1):

1. one being purely differential: $\tilde{A}^{11}z_1' + \tilde{B}^{11}z_1 = \tilde{f}_1$,
2. and the other one being purely algebraic: $\tilde{B}^{22}z_2 = \tilde{f}_2$,

together with some necessary conditions on the right-hand side expressed by $\tilde{f}_3 = 0$. Note that z_3 , when it is present, can be chosen as an arbitrary function.

Finally, we conclude this part by exploring the notion of singularities associated with system (1) (see also [16]).

In the second part of the paper, we extend our approach to handle any higher-order differential system, but for clarity of exposition, we shall describe the reduction method only for second-order systems. As we shall see, the output operator has a very specific form that we shall call *Two-Sided Block-Popov form* since it is similar to the Popov form of Ore matrix polynomials (see [6]). The difference between these two forms is that the former is a square matrix operator obtained by performing elementary operations on rows and columns of the input operator, while the latter is, in general, rectangular computed by working on either rows or columns of the input operator but not on both of them at the same time.

2. NOTATIONS AND TERMINOLOGY

Let $\mathbb{K}$ be a field of characteristic zero, $\mathbb{K}[[x]]$ the ring of formal power series in the variable x and $\mathbb{K}((x))$ its field of fractions. For a matrix $A \in \mathbb{K}((x))^{m \times n}$, $1 \leq i \leq m$ and $1 \leq j \leq n$, we denote by R_i^A its i^{th} row and by C_j^A its j^{th} column. Moreover, we denote by I_n the identity matrix of size n , by $\text{diag}(A^{11}, \dots, A^{nn})$, where A^{ii} are rectangular matrices, the block matrix the i^{th} diagonal block entry of which is A^{ii} and the other block entries are zero matrices, by ∂ the standard derivation $\frac{d}{dx}$ of $\mathbb{K}((x))$ and by $\mathbb{K}((x))[\partial]^{m \times n}$ the ring of $m \times n$ matrix differential operator with coefficients in $\mathbb{K}((x))$. Recall that the multiplication in $\mathbb{K}((x))[\partial]$ is defined as follows: for $a \in \mathbb{K}((x))$, $\partial a = \partial(a) + a\partial$ and we shall use sometimes the notation a' instead of $\partial(a)$. Let $L = \sum_{i=0}^{\ell} A_i(x)\partial^i \in \mathbb{K}((x))[\partial]^{m \times n}$ with nonzero leading coefficient matrix $A_{\ell}(x)$. We then say that L is of *order* ℓ which we denote by $\text{ord}(L) := \ell$. An operator $S \in \mathbb{K}((x))[\partial]^{m \times m}$ ($T \in \mathbb{K}((x))[\partial]^{n \times n}$ respectively) is called *elementary operator* if the multiplication of L on the left by S (on the right by T respectively) consists in one of the following operations:

- multiplying a row (column respectively) of L by a non-zero element of $\mathbb{K}((x))$,
- adding to any row (column respectively) of L another row (column respectively) multiplied by an element of $\mathbb{K}((x))[\partial]$,

- interchanging any two rows (columns respectively) of L .

DEFINITION 2.1. A square matrix differential operator $S \in \mathbb{K}((x))[\partial]^{m \times m}$ is said to be invertible if there exists $\tilde{S} \in \mathbb{K}((x))[\partial]^{m \times m}$ such that $\tilde{S}S = S\tilde{S} = I_m$.

An operator $S \in \mathbb{K}((x))[\partial]^{m \times m}$ is invertible if and only if it can be expressed as a product of elementary operators (see [12]).

DEFINITION 2.2. Two matrix differential operators $L, \tilde{L} \in \mathbb{K}((x))[\partial]^{m \times n}$ are said to be equivalent if there exist two invertible matrix differential operators $S \in \mathbb{K}((x))[\partial]^{m \times m}$ and $T \in \mathbb{K}((x))[\partial]^{n \times n}$ such that $\tilde{L} = SLT$.

When the entries of S and T in Definition 2.2 belong to $\mathbb{K}((x))$, we say that (S, T) is an *algebraic transformation*. Otherwise, we call it a *differential transformation*. Furthermore, transformations of the form (S, I_n) and (I_m, T) will be referred to as *left- and right-transformations* respectively. Two differential systems $L(y) = f$ and $\tilde{L}(z) = \tilde{f}$ are said to be equivalent if the corresponding operators L and $\tilde{L}$ are equivalent and $\tilde{f} = S(f)$, where S is given as in Definition 2.2. In this case, the unknowns y and z are related by $y = T(z)$.

3. REVIEW OF HARRIS' ALGORITHM

In this section, we shall review the algorithm described in the paper [8] of Harris' et al. The aim of our presentation is twofold: we would like to raise awareness of this algorithm, as we have not found references to it within the Computer Algebra community. Secondly, our description of the algorithm as a series of differential and algebraic left- and right-transformations of the input operator $L = A\partial + B \in \mathbb{K}[[x]][\partial]^{m \times n}$ makes it easier to understand the method. This presentation also makes it particularly suitable for an implementation in a Computer Algebra System.

The following lemma is the key element of Harris' algorithm.

LEMMA 3.1. For every matrix $A \in \mathbb{K}[[x]]^{m \times n}$ of rank r , there exist invertible matrices $S \in \mathbb{K}[[x]]^{m \times m}$, $T \in \mathbb{K}[[x]]^{n \times n}$ and $\tilde{A} \in \mathbb{K}[[x]]^{r \times r}$ such that

$$SAT = \begin{bmatrix} \tilde{A} & 0 \\ 0 & 0 \end{bmatrix}.$$

PROOF. The proof is similar to that of [8, Lemma 1]. $\square$

REMARK 3.1. The matrices S, T and $\tilde{A}$ in the above lemma are not necessary invertible at $x = 0$. Furthermore, S and T may be chosen so that $\tilde{A} = x^k I_r$ where $k \in \mathbb{N}$.

In what follows, for ease of presentation, we shall continue using the same symbols L, A, B etc. for the different steps of the algorithm whenever no confusion arises and we denote by $r(L)$ the rank of the leading coefficient A of L , i.e. $r(L) := \text{rank } A$.

3.1 Step 1: Normalisation

The first step of Harris' algorithm is a normalisation step achieved by applying Lemma 3.1 to the leading coefficient A of L . Let $S \in \mathbb{K}[[x]]^{m \times m}$ and $T \in \mathbb{K}[[x]]^{n \times n}$ such that $SAT = \text{diag}(x^k I_r, 0)$ where $k \in \mathbb{N}$ and $r = \text{rank } A$. Thus, by

the algebraic transformation (S, T) , we obtain the equivalent operator

$$SLT = \begin{bmatrix} x^k I_r \partial + B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}. \quad (3)$$

We shall call the form given by (3) a *normalised form*. After computing a normalised form, the algorithm proceeds to Step 2.

3.2 Step 2: Algebraic Reduction

We assume that L is in normalised form as in the r.h.s. of (3). If $B_{22} = 0$, we go to Step 3. Otherwise, we apply Lemma 3.1 to B_{22} and we obtain S, T such that $SB_{22}T = \text{diag}(0, B_{33})$ where B_{33} is an invertible matrix of size $a > 0$. Using the algebraic transformation $(\text{diag}(I_r, S), \text{diag}(I_r, T))$, the new operator can be written as

$$\tilde{L} = \begin{bmatrix} x^k I_r \partial + B_{11} & B_{12} & B_{13} \\ B_{21} & 0 & 0 \\ B_{31} & 0 & B_{33} \end{bmatrix} \in \mathbb{K}[[x]][\partial]^{m \times n}. \quad (4)$$

This latter operator can be further simplified using the algebraic transformation

$$\left(\begin{bmatrix} I_r & & -B_{13}B_{33}^{-1} \\ & I_{m-r-a} & \\ & & I_a \end{bmatrix}, \begin{bmatrix} I_r & & \\ & I_{n-r-a} & \\ -B_{33}^{-1}B_{31} & & I_a \end{bmatrix} \right)$$

eliminating B_{13} and B_{31} in (4). Finally, after multiplying the first row-block by an appropriate power of x , we obtain

$$\begin{bmatrix} x^q I_r \partial + \tilde{B}_{11} & \tilde{B}_{12} & 0 \\ B_{21} & 0 & 0 \\ 0 & 0 & B_{33} \end{bmatrix} := \begin{bmatrix} L_{11} & 0 \\ 0 & B_{33} \end{bmatrix} \quad (5)$$

where $q \in \mathbb{N}$ and $L_{11} \in \mathbb{K}[[x]][\partial]^{(m-a) \times (n-a)}$. The operator L_{11} is in normalised form (3) with $B_{22} = 0$ and we proceed to the next step while the second diagonal block entry of the r.h.s of (5) gives an algebraic system in the new unknown. Note that in this step we have simplified Harris' algorithm by combining Step I (ii) and (iii) in [8].

3.3 Step 3: Differential Row-Reduction

We can now assume that L is of the form

$$L = \begin{bmatrix} x^q I_r \partial + B_{11} & B_{12} \\ B_{21} & 0 \end{bmatrix} \in \mathbb{K}[[x]][\partial]^{m \times n}.$$

If $B_{21} = 0$, then we go to Step 4. Otherwise, we compute S, T such that $SB_{21}T = \text{diag}(0, \tilde{B}_{42})$ where $\tilde{B}_{42}$ is an invertible matrix of size $s > 0$. Write

$$\begin{aligned} \tilde{L} &= \text{diag}(T^{-1}, S) L \text{diag}(T, I_{n-r}) \\ &= \begin{bmatrix} x^q I_{r-s} \partial + \tilde{B}_{11} & \tilde{B}_{12} & \tilde{B}_{13} \\ \tilde{B}_{21} & x^q I_s \partial + \tilde{B}_{22} & \tilde{B}_{23} \\ 0 & 0 & 0 \\ 0 & \tilde{B}_{42} & 0 \end{bmatrix}. \end{aligned}$$

Note that the third row-block of $\tilde{L}$ gives necessary conditions on the right-hand side of the DAE and the fourth results in an algebraic system. Multiplying $\tilde{L}$ on the left by

$$\tilde{S} = \begin{bmatrix} I_{r-s} & & & \\ & I_s & & \\ & & x^q ((\tilde{B}_{42}^{-1})' + \tilde{B}_{42}^{-1} \partial) & \\ & & & I_s \end{bmatrix}$$

will eliminate all the blocks above $\tilde{B}_{42}$. Hence, the operator to consider now is

$$\bar{L} = \begin{bmatrix} x^q I_{r-s} \partial + \tilde{B}_{11} & \tilde{B}_{13} \\ \tilde{B}_{21} & \tilde{B}_{23} \end{bmatrix}$$

which is in normalised form, of smaller size than L and $r(\bar{L}) < r(L)$. The algorithm now proceeds recursively to Step 2.

After successive applications of Steps 2 and 3, either the linear DAE (1) is reduced to a purely algebraic system together with some necessary conditions on the right-hand side or we reach a stage for which there is no algebraic system and we proceed to Step 4.

3.4 Step 4: Differential Column-Reduction

We can assume now that we have an operator of the form

$$L = \begin{bmatrix} x^q I_r \partial + B_{11} & B_{12} \end{bmatrix}. \quad (6)$$

If $B_{12} = 0$, then we find a system of ODEs hence the algorithm is completed. Otherwise, let S, T such that $SB_{12}T = \text{diag}(0, \tilde{B}_{24})$ where $\tilde{B}_{24}$ is an invertible matrix of size $b > 0$. Consider the transformed operator

$$\begin{aligned} \tilde{L} &= S L \text{diag}(S^{-1}, T) \\ &= \begin{bmatrix} x^q I_{r-b} \partial + \tilde{B}_{11} & \tilde{B}_{12} & 0 & 0 \\ \tilde{B}_{21} & x^q I_b \partial + \tilde{B}_{22} & 0 & \tilde{B}_{24} \end{bmatrix} \end{aligned}$$

and define

$$\tilde{T} = \begin{bmatrix} I_{r-b} & & & \\ & I_b & & \\ & & I_{n-r-b} & \\ -\tilde{B}_{24}^{-1} \tilde{B}_{21} & -\tilde{B}_{24}^{-1} (x^q I_b \partial + \tilde{B}_{22}) & & I_b \end{bmatrix}.$$

Multiplying $\tilde{L}$ on the right by $\tilde{T}$ leads to an operator of the form

$$\tilde{L}\tilde{T} = \begin{bmatrix} x^q I_{r-b} \partial + \tilde{B}_{11} & \tilde{B}_{12} & 0 & 0 \\ 0 & 0 & 0 & \tilde{B}_{24} \end{bmatrix},$$

so we consider now

$$\bar{L} = \begin{bmatrix} x^q I_{r-b} \partial + \tilde{B}_{11} & \tilde{B}_{12} \end{bmatrix}$$

which is of the form (6) but of smaller size and $r(\bar{L}) < r(L)$. We repeat this step until we find an operator L of the form (6) for which B_{12} is either invertible or zero matrix, upon which the algorithm terminates. The system (6) can now be solved through either a differential or algebraic system, depending on whether $B_{12} = 0$ or B_{12} is invertible.

4. A NEW REDUCTION ALGORITHM

Motivated by the work of Harris et al, we shall develop a new reduction algorithm which computes an operator of the form (2), equivalent to $L = A\partial + B \in \mathbb{K}[[x]][\partial]^{m \times n}$. This algorithm organises the steps of Harris' algorithm in two main stages: treating the rows of L , and treating its columns. It uses a weaker version of Lemma 3.1 and is essentially based on the computation of left- and right-kernels of rectangular matrices. This is more efficient when implemented in a Computer Algebra System and suitable for a generalisation to higher-order systems.

4.1 Row-Reduction

By means of an algebraic left-transformation, we can assume that

$$A = \begin{bmatrix} A^{11} & A^{12} \\ 0 & 0 \end{bmatrix}$$

where $A^{11} \in \mathbb{K}[[x]]^{r \times r}$ and $r = r(L) = \text{rank } A$. Write $B := (B^{ij})_{i,j=1,2}$ partitioned as in A . Hence, the operator to consider is of the form

$$L = A\partial + B = \begin{bmatrix} A^{11}\partial + B^{11} & A^{12}\partial + B^{12} \\ B^{21} & B^{22} \end{bmatrix}. \quad (7)$$

LEMMA 4.1. *Given a matrix differential operator L of the form (7), assume that*

$$\text{rank} \begin{bmatrix} A^{11} & A^{12} \\ B^{21} & B^{22} \end{bmatrix} < \text{rank} \begin{bmatrix} A^{11} & A^{12} \end{bmatrix} + \text{rank} \begin{bmatrix} B^{21} & B^{22} \end{bmatrix}. \quad (8)$$

Then there exists a differential left-transformation S such that $r(SL) < r(L)$.

PROOF. Equation (8) is equivalent to say that there exists $1 \leq i \leq r$ such that $R_i^A = \sum_{k=1}^r \alpha_k R_k^A + \sum_{j=1}^{m-r} \beta_j R_{r+j}^B$ where $\alpha_k, \beta_j \in K((x))$ and at least one of the β_j is nonzero. Let S be the differential left-transformation defined by the identity matrix of size m the i^{th} row of which is replaced by

$$[-\alpha_1 \quad \cdots \quad 1 \quad \cdots \quad -\alpha_r \quad -\beta_1 \partial \quad \cdots \quad -\beta_{m-r} \partial],$$

where 1 comes at the i^{th} position. Multiplying L on the left by S will replace the i^{th} row of L by

$$\begin{aligned} R_i^A \partial + R_i^B - \sum_{k=1}^r \alpha_k (R_k^A \partial + R_k^B) - \sum_{j=1}^{m-r} \beta_j (\partial(R_{r+j}^B) + R_{r+j}^B \partial) \\ = R_i^B - \sum_{k=1}^r \alpha_k R_k^B - \sum_{j=1}^{m-r} \beta_j \partial(R_{r+j}^B). \end{aligned}$$

This means that we replace the i^{th} row of A by a zero row and that of B by $R_i^B - \sum_{k=1}^r \alpha_k R_k^B - \sum_{j=1}^{m-r} \beta_j \partial(R_{r+j}^B)$, hence $r(SL) < r(L)$. $\square$

We repeat Lemma 4.1 until we find an equivalent operator

$$\tilde{L} = \tilde{A}\partial + \tilde{B} = \begin{bmatrix} \tilde{A}^{11}\partial + \tilde{B}^{11} & \tilde{A}^{12}\partial + \tilde{B}^{12} \\ \tilde{B}^{21} & \tilde{B}^{22} \end{bmatrix}$$

where $\begin{bmatrix} \tilde{A}^{11} & \tilde{A}^{12} \end{bmatrix}$ is of size $q \times n$ ($q = r(\tilde{L}) \leq r(L)$) and has full row rank q with

$$\text{rank} \begin{bmatrix} \tilde{A}^{11} & \tilde{A}^{12} \\ \tilde{B}^{21} & \tilde{B}^{22} \end{bmatrix} = \text{rank} \begin{bmatrix} \tilde{A}^{11} & \tilde{A}^{12} \end{bmatrix} + \text{rank} \begin{bmatrix} \tilde{B}^{21} & \tilde{B}^{22} \end{bmatrix}.$$

The final sub-step of this row-reduction step consists in multiplying $\tilde{L}$ by an algebraic left-transformation to eliminate all the linearly dependent rows of $\begin{bmatrix} \tilde{B}^{21} & \tilde{B}^{22} \end{bmatrix}$.

4.2 Column-Reduction

Let L be a matrix differential operator of the form (7) where $\begin{bmatrix} A^{11} & A^{12} \end{bmatrix}$ is of size $q \times n$ and $r(L) = q$. Let T_1 be an invertible matrix such that

$$\tilde{A} := AT_1 = \begin{bmatrix} \tilde{A}^{11} & 0 \\ 0 & 0 \end{bmatrix},$$

where $\tilde{A}^{11}$ is an invertible matrix of size q and denote $\tilde{B} = (\tilde{B}^{ij})_{i,j=1,2} := AT_1' + BT_1$. Hence, L is equivalent to

$$\tilde{L} = LT_1 = \begin{bmatrix} \tilde{A}^{11}\partial + \tilde{B}^{11} & \tilde{B}^{12} \\ \tilde{B}^{21} & \tilde{B}^{22} \end{bmatrix}. \quad (9)$$

The following lemma can be proved similarly to Lemma 4.1.

LEMMA 4.2. *Given a matrix differential operator $\tilde{L}$ of the form (9), we assume that*

$$\text{rank} \begin{bmatrix} \tilde{A}^{11} & \tilde{B}^{12} \\ 0 & \tilde{B}^{22} \end{bmatrix} < \text{rank } \tilde{A}^{11} + \text{rank} \begin{bmatrix} \tilde{B}^{12} \\ \tilde{B}^{22} \end{bmatrix}.$$

Then there exists a differential right-transformation T_2 such that $r(\tilde{L}T_2) < r(\tilde{L})$.

We repeat the application of Lemma 4.2 until we find an equivalent operator

$$\bar{L} = \bar{A}\partial + \bar{B} = \begin{bmatrix} \bar{A}^{11}\partial + \bar{B}^{11} & \bar{B}^{12} \\ \bar{B}^{21} & \bar{B}^{22} \end{bmatrix}$$

where $\bar{A}^{11}$ is of size $q \times r(\bar{L})$ ($r(\bar{L}) \leq q = r(\tilde{L})$) and has full column rank verifying

$$\text{rank} \begin{bmatrix} \bar{A}^{11} & \bar{B}^{12} \\ 0 & \bar{B}^{22} \end{bmatrix} = \text{rank } \bar{A}^{11} + \text{rank} \begin{bmatrix} \bar{B}^{12} \\ \bar{B}^{22} \end{bmatrix}.$$

Finally, we multiply $\bar{L}$ by an algebraic right-transformation to cancel all linearly dependent columns of $\begin{bmatrix} \bar{B}^{12} \\ \bar{B}^{22} \end{bmatrix}$.

4.3 Decoupling Differential and Algebraic Equations

After carrying out a series of row- and column-reductions until neither Lemma 4.1 nor Lemma 4.2 can be applied, the operator L given by (1) will be equivalent to

$$\tilde{L} = \begin{bmatrix} \tilde{A}^{11}\partial + \tilde{B}^{11} & \tilde{B}^{12} & 0 \\ \tilde{B}^{21} & \tilde{B}^{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} := \begin{bmatrix} \tilde{L}_1 & 0 \\ 0 & 0 \end{bmatrix}$$

where $\tilde{L}_1$ is of size $p \times s$ and $\tilde{A}^{11}$ is an invertible matrix of size d . Furthermore, we have

$$\text{rank} \begin{bmatrix} \tilde{A}^{11} & 0 \\ \tilde{B}^{21} & \tilde{B}^{22} \end{bmatrix} = p \quad (10)$$

and

$$\text{rank} \begin{bmatrix} \tilde{A}^{11} & \tilde{B}^{12} \\ 0 & \tilde{B}^{22} \end{bmatrix} = s. \quad (11)$$

Since $\tilde{A}^{11}$ is invertible, equation (10) (equation (11) respectively) implies that $\tilde{B}^{22}$ has full row rank (full column rank respectively). Consequently $\tilde{B}^{22}$ is invertible and $p = s$. Now it is easy to decouple the operator $\tilde{L}_1$ into a purely differential and purely algebraic system. Indeed,

$$\begin{bmatrix} I_d & -\tilde{B}^{12}(\tilde{B}^{22})^{-1} \\ 0 & I_{p-d} \end{bmatrix} \tilde{L}_1 \begin{bmatrix} I_d & 0 \\ -(\tilde{B}^{22})^{-1}\tilde{B}^{21} & I_{p-d} \end{bmatrix} = \begin{bmatrix} \tilde{A}^{11}\partial + \tilde{B}^{11} - \tilde{B}^{12}(\tilde{B}^{22})^{-1}\tilde{B}^{21} & 0 \\ 0 & \tilde{B}^{22} \end{bmatrix}.$$

Multiplying this latter system by a suitable power of x , we can assume that its entries belong to $\mathbb{K}[[x]][\partial]$. This leads to the following

THEOREM 4.1. *Given a linear DAE of the form (1), there exist two invertible operators $S \in \mathbb{K}((x))[\partial]^{m \times m}$ and $T \in \mathbb{K}((x))[\partial]^{n \times n}$ that transform (1) to a decoupled system of the form*

$$\tilde{L}(z) = \begin{bmatrix} \tilde{A}^{11}\partial + \tilde{B}^{11} & 0 & 0 \\ 0 & \tilde{B}^{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} \tilde{f}_1 \\ \tilde{f}_2 \\ \tilde{f}_3 \end{bmatrix} \quad (12)$$

where $\tilde{L} \in \mathbb{K}[[x]][\partial]^{m \times n}$, $\tilde{A}^{11}$ and $\tilde{B}^{22}$ are both invertible, $y = T(z)$ and $\tilde{f} = S(f)$.

REMARK 4.1. If $\tilde{f}_3 = 0$ in (12), then the system is said to be consistent and admits at least one solution. If moreover $\text{rank}(\tilde{A}^{11}) + \text{rank}(\tilde{B}^{22}) < n$ then z_3 can be chosen as arbitrary function hence the dimension of the affine solution space of (12) is infinite.

REMARK 4.2. We could replace $\tilde{A}^{11}$ and $\tilde{B}^{22}$ in (12) by identity matrices, but this adds computational overhead and is not necessarily required. For example, if one wants to compute regular solutions of $\tilde{L}(z) = 0$ where $\tilde{L}$ is given by (12), the algorithm developed in [3] handles directly the system $\tilde{A}^{11}z_1' + \tilde{B}^{11}z_1 = 0$ where $\tilde{A}^{11}$ can be of arbitrary form.

REMARK 4.3. By carrying out further reductions on the operator $\tilde{L}$ given by (12), one can obtain a Jacobson form (see [11]) of L given by (1). This latter form requires computing cyclic vectors and yields a scalar differential operator with very large coefficients especially when the systems have large size (see e.g. [1]). Hence, from an algorithmic point of view, it is better to manipulate the systems directly than to convert them to scalar differential equations.

EXAMPLE 4.1. We consider the linear DAE

$$L(y) = A(x)y' + B(x)y = f(x)$$

where $A(x)$ and $B(x)$ are given respectively by the matrices

$$\begin{bmatrix} 1 & x & 1 & -x \\ x^2 & 2+x & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1+x & -x \\ 0 & 0 & -1 & x \end{bmatrix}$$

and $f(x)$ is an arbitrary vector of size 3. Following the steps of our algorithm, we find

$$S = \begin{bmatrix} -x^2 & 1 & \frac{-7x-2x^2-2+x^3}{-2-x+x^3} \\ -2-x+x^3 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$T = \begin{bmatrix} -x & \frac{2+x}{-2-x+x^3} & 0 & -1 \\ 1 & \frac{-x^2}{-2-x+x^3} & 0 & 0 \\ x & 1 & x & x\partial \\ 1 & 0 & 1 & \partial \end{bmatrix}$$

and the decoupled operator $\tilde{L} = SLT$ is given by

$$\tilde{L} = \begin{bmatrix} (-x^3+x+2)\partial & 0 & 0 & 0 \\ 0 & x^2 & x^3-x-2 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}.$$

Remark that this system is consistent for any right-hand side $f(x)$.

4.4 Application: Classification of Singularities

It is well known that, in the classical theory of ODEs (see e.g. [18]), a singularity is classified as *regular* or *irregular*. Our algorithm makes possible to extend this notion to linear DAEs since it reduces this problem to the ODE case.

DEFINITION 4.1. *For a homogeneous linear DAE of the form (1), the origin is called a regular singularity, if it is a regular singularity of the ODE $\tilde{A}^{11}z_1' + \tilde{B}^{11}z_1 = 0$ and $\text{rank}(\tilde{A}^{11}) + \text{rank}(\tilde{B}^{22}) = n$ where $\tilde{A}^{11}$, $\tilde{B}^{11}$ and $\tilde{B}^{22}$ are given by (12). Otherwise, it is called an irregular singularity.*

Consequently, we are able to algorithmically decide whether a given homogeneous linear DAE is regular or irregular singular at $x = 0$ by applying our reduction algorithm in order to compute the decoupled system (12) and using techniques developed for the ODE case (e.g. computing a *Moser-irreducible* form [4, 13] of $z_1' + (\tilde{A}^{11})^{-1}\tilde{B}^{11}z_1 = 0$). However, it is currently an open problem how to algorithmically classify singularities without this decoupling.

5. GENERALISATION TO SECOND-ORDER SYSTEMS

In this section, we would like to generalise our approach to handle higher-order systems, but for reasons of clarity, we restrict our study to linear DAEs of second-order of the form

$$L(y) = A\partial^2 y + B\partial y + Cy = f \quad (13)$$

where $A, B, C \in \mathbb{K}[[x]]^{m \times n}$ ($A \neq 0$) and $f \in \mathbb{K}[[x]]^m$. The first method that comes to mind is to transform (13) to a first-order linear DAE then to apply the procedure presented in Section 4. But, this is not very desirable since it increases the size of the system and can break the structure of the first-order system. To circumvent that, we shall develop a method that handles directly linear second-order systems. The goal is to reduce (13) to an equivalent system of simpler form and possibly lower size using the same type of reductions as in the first-order case. As we shall see, this method will decouple (13) into an algebraic system and a square second-order DAE of lower size and full rank (see [5, Def 2.1]) having a specific form that we shall call *Two-Sided Block-Popov form*.

DEFINITION 5.1. *Let $L = (L_{ij})_{i,j=1,\dots,k}$ be a square linear matrix differential operator of size $n \times n$, where $L_{ij} \in \mathbb{K}[[x]][\partial]^{n_i \times n_j}$ and $n = \sum_{i=1}^k n_i$. We say that L is in Two-Sided Block-Popov form if, for every $i \in \{1, \dots, k\}$, we have*

- $\text{ord}(L_{ii}) > \text{ord}(L_{jj})$ if $i < j$,
- $\text{ord}(L_{ij}) < \text{ord}(L_{ii}) \forall j \neq i$,
- $\text{ord}(L_{ji}) < \text{ord}(L_{ii}) \forall j \neq i$,
- the leading coefficient of L_{ii} is invertible in $\mathbb{K}((x))^{n_i \times n_i}$.

At the end of this section, we shall show how a second-order DAE in Two-Sided Block-Popov form can be transformed into a first-order system of ODEs. But first, we shall describe our reduction method which is based on successive applications of row- and column-reductions until we find an equivalent operator verifying some properties allowing the decoupling to take place.

5.1 Row-Reduction

Let $S_1 \in \mathbb{K}[[x]]^{m \times m}$ be an invertible matrix such that

$$S_1 A := \begin{bmatrix} A^{11} & A^{12} & A^{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (14)$$

where $\begin{bmatrix} A^{11} & A^{12} & A^{13} \end{bmatrix}$ is of size $r \times n$ and rank r . Write $S_1 B := (B^{ij})_{i,j=1,\dots,3}$ partitioned as $S_1 A$. Now, let $S_2 \in \mathbb{K}[[x]]^{(m-r) \times (m-r)}$ be an invertible matrix such that

$$\text{diag}(I_r, S_2) S_1 B := \begin{bmatrix} B^{11} & B^{12} & B^{13} \\ B^{21} & B^{22} & B^{23} \\ 0 & 0 & 0 \end{bmatrix} \quad (15)$$

where $\begin{bmatrix} B^{21} & B^{22} & B^{23} \end{bmatrix}$ is of size $k \times n$ and rank k and finally, write $\text{diag}(I_r, S_2) S_1 C = (C^{ij})_{i,j=1,\dots,3}$ partitioned as in (14) and (15). Consequently, by means of algebraic left-transformations, L can be supposed to be of the form (13) where A and B are respectively given by (14) and (15). To refer to this form in the sequel, we shall call it a *normalised row form* and denote by $r(L) := r$, $k(L) := k$ and by $\mathcal{M}(L)$ and $\mathcal{N}(L)$ respectively the matrices

$$\begin{bmatrix} A^{11} & A^{12} & A^{13} \\ B^{21} & B^{22} & B^{23} \\ C^{31} & C^{32} & C^{33} \end{bmatrix} \text{ and } \begin{bmatrix} B^{21} & B^{22} & B^{23} \\ C^{31} & C^{32} & C^{33} \end{bmatrix}.$$

Step R.1. We suppose that L is in normalised row form and a row of $\begin{bmatrix} A^{11} & A^{12} & A^{13} \end{bmatrix}$ is a linear combination of the other rows of $\begin{bmatrix} A^{11} & A^{12} & A^{13} \end{bmatrix}$ and at least one row of $\mathcal{N}(L)$. Then, there exists a differential left-transformation S such that $r(SL) < r(L)$. Indeed, let $1 \leq i \leq r = r(L)$ such that

$$R_i^A + \sum_{\substack{j=1 \\ j \neq i}}^r \alpha_j R_j^A + \sum_{p=1}^k \beta_p R_{p+r}^B + \sum_{s=1}^{m-r-k} \gamma_s R_{r+k+s}^C = 0,$$

where $\alpha_j, \beta_p, \gamma_s \in \mathbb{K}((x))$ and β_p, γ_s are not all zero. Define S as the identity matrix of size m the i^{th} row of which is replaced by the row formed by the coefficients of the above linear combination in which we replace β_p by $\beta_p \partial$ for $p = 1, \dots, k$ and γ_s by $\gamma_s \partial^2$ for $s = 1, \dots, m-r-k$ (S is similar to that in the proof of Lemma 4.1 but here we have also the terms $\gamma_s \partial^2$). The multiplication of L on the left by S can be seen as follows:

- R_i^A is replaced by zero row,
- R_i^B is replaced by $R_i^B + \sum_{\substack{j=1 \\ j \neq i}}^r \alpha_j R_j^B + \sum_{p=1}^k \beta_p (\partial(R_{r+p}^B) + R_{r+p}^C) + 2 \sum_{s=1}^{m-r-k} \gamma_s \partial(R_{r+k+s}^C)$,
- R_i^C is replaced by $R_i^C + \sum_{\substack{j=1 \\ j \neq i}}^r \alpha_j R_j^C + \sum_{p=1}^k \beta_p \partial(R_{r+p}^C) + \sum_{s=1}^{m-r-k} \gamma_s \partial^2(R_{r+k+s}^C)$.

Consequently, we have $r(SL) < r(L)$ and $k(SL) \leq k(L) + 1$.

We repeat this step until we find an equivalent operator $\tilde{L} = \tilde{A} \partial^2 + \tilde{B} \partial + \tilde{C}$ in normalised row form verifying the equality

$$\text{rank } \mathcal{M}(\tilde{L}) = \text{rank } \begin{bmatrix} \tilde{A}^{11} & \tilde{A}^{12} & \tilde{A}^{13} \end{bmatrix} + \text{rank } \mathcal{N}(\tilde{L}), \quad (16)$$

then we proceed to Step R.2.

Step R.2. We assume that $\tilde{L} = \tilde{A} \partial^2 + \tilde{B} \partial + \tilde{C}$ is in normalised row form and the operator

$$\begin{bmatrix} \tilde{B}^{21} \partial + \tilde{C}^{21} & \tilde{B}^{22} \partial + \tilde{C}^{22} & \tilde{B}^{23} \partial + \tilde{C}^{23} \\ \tilde{C}^{31} & \tilde{C}^{32} & \tilde{C}^{33} \end{bmatrix}$$

satisfies the assumption of Lemma 4.1. Then there exists a differential left-transformation S such that $k(S\tilde{L}) < k(\tilde{L})$ and $r(S\tilde{L}) = r(\tilde{L})$.

We repeat this until we obtain an equivalent operator $\hat{L} = \hat{A} \partial^2 + \hat{B} \partial + \hat{C}$ in normalised row form verifying

$$\text{rank } \mathcal{N}(\hat{L}) = \text{rank } \begin{bmatrix} \hat{B}^{21} & \hat{B}^{22} & \hat{B}^{23} \end{bmatrix} + \text{rank } \begin{bmatrix} \hat{C}^{31} & \hat{C}^{32} & \hat{C}^{33} \end{bmatrix}. \quad (17)$$

But this does not necessarily mean that $\hat{L}$ verifies also equation (16). If this is the case, we multiply $\hat{L}$ on the left by an algebraic transformation to annihilate all dependent rows of $\begin{bmatrix} \hat{C}^{31} & \hat{C}^{32} & \hat{C}^{33} \end{bmatrix}$ and we are done. Otherwise, we go back to Step R.1. Consequently, after successive applications of Steps R.1 and R.2, the number of differential equations of second-order or the total number of differential equations decrease. Hence it is assured that eventually we shall obtain an equivalent operator $\bar{L}$ in normalised row form for which equations (16) and (17) do hold.

5.2 Column-Reduction

By means of algebraic right-transformations applied on L given by (13), we can suppose that A and B are respectively of the form

$$A := \begin{bmatrix} A^{11} & 0 & 0 \\ A^{21} & 0 & 0 \\ A^{31} & 0 & 0 \end{bmatrix} \text{ and } B := \begin{bmatrix} B^{11} & B^{12} & 0 \\ B^{21} & B^{22} & 0 \\ B^{31} & B^{32} & 0 \end{bmatrix},$$

where $\begin{bmatrix} A^{11} \\ A^{21} \\ A^{31} \end{bmatrix}$ (the matrix $\begin{bmatrix} B^{12} \\ B^{22} \\ B^{32} \end{bmatrix}$ respectively) is of size $m \times r$ and rank r (of size $m \times s$ and rank s respectively). Write $C := (C^{ij})_{i,j=1,\dots,3}$ of the same partition as A and B . The operator L of this form is said to be in *normalised column form* with which we associate $r(L) := r$, $s(L) := s$,

$$\mathcal{P}(L) := \begin{bmatrix} A^{11} & B^{12} & C^{13} \\ A^{21} & B^{22} & C^{23} \\ A^{31} & B^{32} & C^{33} \end{bmatrix} \text{ and } \mathcal{Q}(L) := \begin{bmatrix} B^{12} & C^{13} \\ B^{22} & C^{23} \\ B^{32} & C^{33} \end{bmatrix}.$$

Here we shall proceed in the same way as in row-reduction in order to find an equivalent operator $\bar{L}$ such that all the nonzero columns of $\mathcal{P}(\bar{L})$ are linearly independent.

Step C.1. We suppose that L is in normalised column form

and a column of the matrix $\begin{bmatrix} A^{11} \\ A^{21} \\ A^{31} \end{bmatrix}$ is a linear combination

of its other columns and at least one column of $\mathcal{Q}(L)$. Then, there exists a differential right-transformation T such that $r(LT) < r(L)$. Indeed, let $1 \leq i \leq r = r(L)$ s.t.

$$C_i^A + \sum_{\substack{j=1 \\ j \neq i}}^r \alpha_j C_j^A + \sum_{p=1}^s \beta_p C_{p+r}^B + \sum_{d=1}^{n-r-s} \gamma_d C_{r+s+d}^C = 0,$$

where $\alpha_j, \beta_p, \gamma_d \in \mathbb{K}((x))$ and β_p, γ_d are not all zero. Define T as the identity matrix of size n the i^{th} column of which is

replaced by the column vector formed by the coefficients of the above linear combination in which we replace β_p by $\beta_p \partial$ for $p = 1, \dots, s$ and γ_d by $\gamma_d \partial^2$ for $d = 1, \dots, n - r - s$. The multiplication of L on the right by T can be seen as follows:

- C_i^A is replaced by zero column;
- C_i^B is replaced by $C_i^B + \sum_{j=1}^r (2\partial(\alpha_j)C_j^A + \alpha_j C_j^B) + \sum_{p=1}^s (\partial(\beta_p)C_{r+p}^B + \beta_p C_{r+p}^C)$;
- C_i^C is replaced by $C_i^C + \sum_{j=1}^r (\partial^2(\alpha_j)C_j^A + \partial(\alpha_j)C_j^B + \alpha_j C_j^C)$.

Consequently, we have $r(LT) < r(L)$ and $s(LT) \leq s(L) + 1$.

We repeat this step until we find an equivalent operator $\tilde{L} = \tilde{A}\partial^2 + \tilde{B}\partial + \tilde{C}$ in normalised column form verifying the equality

$$\text{rank } \mathcal{P}(\tilde{L}) = \text{rank } \tilde{A} + \text{rank } \mathcal{Q}(\tilde{L}). \quad (18)$$

Then we proceed to Step C.2.

Step C.2. We assume that $\tilde{L} = \tilde{A}\partial^2 + \tilde{B}\partial + \tilde{C}$ is in normalised column form and the operator

$$\begin{bmatrix} \tilde{B}^{12}\partial + \tilde{C}^{12} & \tilde{C}^{13} \\ \tilde{B}^{22}\partial + \tilde{C}^{22} & \tilde{C}^{23} \\ \tilde{B}^{32}\partial + \tilde{C}^{32} & \tilde{C}^{33} \end{bmatrix}$$

satisfies the assumptions of Lemma 4.2. Then there exists a differential right-transformation T such that $s(\tilde{L}T) < s(\tilde{L})$ and $r(\tilde{L}T) = r(\tilde{L})$.

We repeat this until we obtain an equivalent operator $\hat{L} = \hat{A}\partial^2 + \hat{B}\partial + \hat{C}$ in normalised column form verifying

$$\text{rank } \mathcal{Q}(\hat{L}) = \text{rank} \begin{bmatrix} \hat{B}^{12} \\ \hat{B}^{22} \\ \hat{B}^{32} \end{bmatrix} + \text{rank} \begin{bmatrix} \hat{C}^{13} \\ \hat{C}^{23} \\ \hat{C}^{33} \end{bmatrix}. \quad (19)$$

Note that here again $\hat{L}$ may not verify equation (18) so we have to go back to Step C.1. Consequently, after number of iterations of Steps C.1 and C.2, we obtain an equivalent operator $\bar{L}$ in normalised column form verifying equations (18) and (19). Finally, multiply $\bar{L}$ on the right by an algebraic transformation to annihilate all dependent columns of the sub-matrix $(\bar{C}^{i3})_{i=1,\dots,3}$.

5.3 Computing Two-Sided Block-Popov Forms

We start with an operator L of the form (13) with associated quantities $r(L)$, $k(L)$ and $s(L)$ defined as above. In order to obtain an equivalent operator in Two-Sided Block-Popov form, we recursively apply to L a series of row- and column-reductions until the triplet $(r(L), k(L), s(L))$ becomes minimal in the sense of the lexicographic ordering. This algorithm terminates because at each step, this triplet decreases. Indeed, at each step of the algorithm, we either perform

- Step R.1. or Step C.1, in which case, $r(L)$ will decrease (however the quantities $k(L)$ or $s(L)$ may increase), or
- Step R.2. in which case, $r(L)$ remains unchanged and $k(L)$ will decrease ($s(L)$ may decrease), or
- Step C.2. in which case, $r(L)$ remains unchanged, $k(L)$ cannot increase and $s(L)$ will decrease.

Hence this algorithm returns an operator equivalent to L of the form

$$\tilde{L} = \begin{bmatrix} \tilde{L}^{11} & 0 \\ 0 & 0 \end{bmatrix}$$

where

$$\tilde{L}^{11} = \begin{bmatrix} \tilde{A}^{11}\partial^2 + \tilde{B}^{11}\partial + \tilde{C}^{11} & \tilde{B}^{12}\partial + \tilde{C}^{12} & \tilde{C}^{13} \\ \tilde{B}^{21}\partial + \tilde{C}^{21} & \tilde{B}^{22}\partial + \tilde{C}^{22} & \tilde{C}^{23} \\ \tilde{C}^{31} & \tilde{C}^{32} & \tilde{C}^{33} \end{bmatrix},$$

$\tilde{A}^{11}$ is invertible and $\mathcal{M}(\tilde{L}^{11})$ and $\mathcal{P}(\tilde{L}^{11})$ given respectively by

$$\begin{bmatrix} \tilde{A}^{11} & 0 & 0 \\ \tilde{B}^{21} & \tilde{B}^{22} & 0 \\ \tilde{C}^{31} & \tilde{C}^{32} & \tilde{C}^{33} \end{bmatrix} \text{ and } \begin{bmatrix} \tilde{A}^{11} & \tilde{B}^{12} & \tilde{C}^{13} \\ 0 & \tilde{B}^{22} & \tilde{C}^{23} \\ 0 & 0 & \tilde{C}^{33} \end{bmatrix}$$

have respectively full row rank and full column rank. Since $\tilde{A}^{11}$ is invertible, this implies that the matrices

$$\begin{bmatrix} \tilde{A}^{11} & 0 & 0 \\ 0 & \tilde{B}^{22} & 0 \\ 0 & \tilde{C}^{32} & \tilde{C}^{33} \end{bmatrix} \text{ and } \begin{bmatrix} \tilde{A}^{11} & 0 & 0 \\ 0 & \tilde{B}^{22} & \tilde{C}^{23} \\ 0 & 0 & \tilde{C}^{33} \end{bmatrix}$$

have respectively full row rank and full column rank, hence $\tilde{B}^{22}$ is invertible. In the same way, we show that $\tilde{C}^{33}$ is also invertible. Consequently, by means of algebraic transformation, we can eliminate the matrices $\tilde{B}^{21}$, $\tilde{B}^{12}$, $\tilde{C}^{31}$, $\tilde{C}^{32}$, $\tilde{C}^{13}$ and $\tilde{C}^{23}$ in $\tilde{L}^{11}$. Hence this latter operator is equivalent to $\text{diag}(\hat{L}, \tilde{C}^{33})$ where

$$\hat{L} := \begin{bmatrix} \tilde{A}^{11}\partial^2 + \tilde{B}^{11}\partial + \tilde{C}^{11} & \hat{C}^{12} \\ \tilde{C}^{21} & \tilde{B}^{22}\partial + \tilde{C}^{22} \end{bmatrix}$$

is a square matrix differential operator of full rank and in Two-Sided Block-Popov form. We have hence shown the following theorem:

THEOREM 5.1. *For every linear differential system of second-order of the form (13), there exists two invertible operators $S \in \mathbb{K}((x))[\partial]^{m \times m}$ and $T \in \mathbb{K}((x))[\partial]^{n \times n}$ that transform (13) to the decoupled system $\hat{L}(z) = \hat{f}$ where $\hat{L} \in \mathbb{K}[[x]][\partial]^{m \times n}$ is of the form*

$$\hat{L} := \begin{bmatrix} \tilde{A}^{11}\partial^2 + \tilde{B}^{11}\partial + \tilde{C}^{11} & \tilde{C}^{12} & 0 & 0 \\ \tilde{C}^{21} & \tilde{B}^{22}\partial + \tilde{C}^{22} & 0 & 0 \\ 0 & 0 & \tilde{C}^{33} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad (20)$$

$\tilde{A}^{11}$, $\tilde{B}^{22}$ and $\tilde{C}^{33}$ are invertible matrices, $y = T(z)$ and $\hat{f} = S(f)$.

5.4 Applications

In this section, we consider homogeneous second-order system of the form

$$L(y) = A\partial^2 y + B\partial y + Cy = 0, \quad (21)$$

where $0 \neq A \in \mathbb{K}[[x]]^{m \times n}$ and $B, C \in \mathbb{K}[[x]]^{m \times n}$. Theorem 5.1 shows that it is equivalent to $\hat{L}(z) = 0$ where $\hat{L}$ is given by (20). Furthermore, the second-order DAE in Two-Sided Block-Popov form

$$\begin{bmatrix} \tilde{A}^{11}\partial^2 + \tilde{B}^{11}\partial + \tilde{C}^{11} & \tilde{C}^{12} \\ \tilde{C}^{21} & \tilde{B}^{22}\partial + \tilde{C}^{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = 0 \quad (22)$$

can be converted into the first-order system of ODEs

$$\left(\begin{bmatrix} I & & \\ & \tilde{A}^{11} & \\ & & \tilde{B}^{22} \end{bmatrix} \partial + \begin{bmatrix} 0 & -I & 0 \\ \tilde{C}^{11} & \tilde{B}^{11} & \tilde{C}^{12} \\ \tilde{C}^{21} & 0 & \tilde{C}^{22} \end{bmatrix} \right) \begin{bmatrix} z_1 \\ z_1' \\ z_2 \end{bmatrix} = 0. \quad (23)$$

This will allow us to extend some notions known for the ODE to the DAE case.

5.4.1 Classification of singularities

Here again, we call the origin a *regular singularity* of (21) if it is for the first-order system given by (23) and $\text{rank}(\tilde{A}^{11}) + \text{rank}(\tilde{B}^{22}) + \text{rank}(\tilde{C}^{33}) = n$ where $\tilde{A}^{11}$, $\tilde{B}^{22}$, $\tilde{C}^{33}$ are given as in (20). Otherwise, it is called an *irregular singularity*.

5.4.2 Regular Solutions

Suppose that $\tilde{L}$ given by (20) satisfies

$$\text{rank}(\tilde{A}^{11}) + \text{rank}(\tilde{B}^{22}) + \text{rank}(\tilde{C}^{33}) = n,$$

which is equivalent to say that the dimension of the solution space of $\tilde{L}(z) = 0$ is finite. We are interested in computing a basis of the regular solution space of $L(y) = 0$. Recall that a regular solution is a solution of the form $y = x^{\lambda_0} w$ where $\lambda_0 \in \mathbb{K}$, $w \in \mathbb{K}[[x]][\log x]^n$ and $\mathbb{K}$ denotes the algebraic closure of $\mathbb{K}$. To our knowledge, there is currently no method that directly handles systems of the form (21). Our algorithm reduces this problem to the computation of a basis of the regular solution space of the DAE in Two-Sided Block-Popov form given by (22) (since $\tilde{C}^{33}$ is invertible) which is equivalent to solve system (23). But there exists a direct and simpler method to solve (22). Indeed, let $p(\lambda)$ denote the determinant of the following matrix polynomial

$$\begin{bmatrix} \tilde{A}^{11}(0)(\lambda^2 - \lambda) + \tilde{B}^{11}(0)\lambda + \tilde{C}^{11}(0) & \tilde{C}^{12}(0) \\ \tilde{C}^{21}(0) & \tilde{B}^{22}(0)\lambda + \tilde{C}^{22}(0) \end{bmatrix}.$$

The simplest situation is when $p(\lambda) \not\equiv 0$. In this case, we can apply the algorithm described in [2] and show that the dimension of this space is exactly equal to $\deg(p(\lambda))$. Hence, if we denote by δ and γ respectively the size of the matrices $\tilde{A}^{11}$ and $\tilde{B}^{22}$, then the origin is a regular singularity of (21) if and only if $\deg(p(\lambda)) = 2\delta + \gamma$ which is equivalent to say that $\tilde{A}^{11}(0)$ and $\tilde{B}^{22}(0)$ are invertible. The other situation ($p(\lambda) \equiv 0$) is currently being studied by us, in which the fact that (22) has full rank plays an important role.

6. CONCLUSION

In this paper, we have developed a new reduction algorithm that reduces a linear DAE to a decoupled differential and algebraic system. We have extended our approach to handle second-order systems. Our algorithm for first-order DAEs has been implemented in the Computer Algebra system Maple as a part of the ISOLDE Package. It seems to be efficient for small examples and is currently being tested and improved to obtain a better performance when handling systems of bigger size.

7. REFERENCES

- [1] M. Barkatou. On rational solutions of systems of linear differential equations. *J. of Symbolic Computation*, 28:547-567, 1999.
- [2] M. Barkatou, T. Cluzeau and C. El Bacha. Algorithms for regular solutions of higher-order linear differential

systems. In *Proceedings of ISSAC'09*, pages 7-14, Seoul, South Korea, 2009. ACM.

- [3] M. Barkatou and E. Pflügel. An algorithm computing the regular formal solutions of a system of linear differential equations. *J. of Symbolic Computation*, 28:569-587, 1999.
- [4] M. Barkatou and E. Pflügel. On the Moser- and super-reduction algorithms of systems of linear differential equations and their complexity. *J. of Symbolic Computation*, 44(8):1017-1036, 2009.
- [5] B. Beckermann, H. Cheng and G. Labahn. Fraction-free row reduction of matrices of Ore polynomials. *J. of Symbolic Computation*, 41(5):513-543, 2006.
- [6] P. Davies, H. Cheng and G. Labahn. Computing Popov form of general Ore polynomial matrices. In *Milestones in Computer Algebra*, pages 149-156, 2008.
- [7] C. W. Gear. Differential-algebraic equation index transformation. *SIAM J. Sci. Stat. Comput.*, 9(1):39-47, 1988.
- [8] W. A. Harris, Y. Sibuya and L. Weinberg. A reduction algorithm for linear differential systems. *Funkcialaj Ekvacioj*, 11:59-67, 1968.
- [9] P. Kunkel and V. Mehrmann. Canonical forms for linear differential-algebraic equations with variable coefficients. *J. of Computational and Applied Mathematics*, 56:225-251, 1994.
- [10] V. Mehrmann and C. Shi. Transformation of higher order differential-algebraic systems to first order. *Numerical Algorithms*, 42:281-307, 2006.
- [11] J. Middeke. A polynomial-time algorithm for the Jacobson form for matrices of differential operators. Tech. report no. 08-13 in RISC Report Series, 2008.
- [12] M. Miyake. Remarks on the formulation of the Cauchy problem for general system of ordinary differential equations. *Tôhoku Math. J.*, 32:79-89, 1980.
- [13] J. Moser. The order of a singularity in Fuchs' theory. *Math. Zeitschr.*, 72:379-398, 1960.
- [14] A. Pantelous, A. Karageorgos and G. Kalogeropoulos. Power series solutions for linear higher order rectangular differential matrix control systems. In *17th Mediterranean Conference on Control and Automation*, p. 330-335, Makedonia Palace, Thessaloniki, Greece, 2009.
- [15] M. P. Quéré and G. Villard. An algorithm for the reduction of linear DAE. In *Proceedings of ISSAC'95*, pages 223-231, New York, USA, 1995. ACM.
- [16] M. P. Quéré-Stuchlik. Algorithmique des faisceaux linéaires de matrices, Applications à la théorie des systèmes linéaires et à la résolution d'équations algébro-différentielles. *PhD thesis*, LMC-IMAG, 1996.
- [17] S. Schulz. Four lectures on differential algebraic equations. Tech. Report 497, The University of Auckland, 2003.
- [18] W. Wasow. Asymptotic expansions for ordinary differential equations. Robert E. Krieger Publ., 1967.

Consistency of Finite Difference Approximations for Linear PDE Systems and Its Algorithmic Verification

Vladimir P. Gerdt
Laboratory of Information Technologies,
Joint Institute for Nuclear Research
141980 Dubna, Russia
gerdt@jinr.ru

Daniel Robertz
Lehrstuhl B für Mathematik,
RWTH Aachen University
Templergraben 64, 52056 Aachen, Germany
daniel@momo.math.rwth-aachen.de

ABSTRACT

In this paper we consider finite difference approximations for numerical solving of systems of partial differential equations of the form $f_1 = \dots = f_p = 0$, where $F := \{f_1, \dots, f_p\}$ is a set of linear partial differential polynomials over the field of rational functions with rational coefficients. For orthogonal and uniform solution grids we strengthen the generally accepted concept of equation-wise consistency (e-consistency) of the difference equations $\tilde{f}_1 = \dots = \tilde{f}_p = 0$ as approximation of the differential ones. Instead, we introduce a notion of consistency of the set of all linear consequences of the difference polynomial set $\tilde{F} := \{\tilde{f}_1, \dots, \tilde{f}_p\}$ with the linear subset of the differential ideal $\langle F \rangle$. The last consistency, which we call s-consistency (strong consistency), admits algorithmic verification via a Gröbner basis of the difference ideal $\langle \tilde{F} \rangle$. Some related illustrative examples of finite difference approximations, including those which are e-consistent and s-inconsistent, are given.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms; I.1.4 [Symbolic and Algebraic Manipulation]: Applications

General Terms

Algorithms, Applications

Keywords

Partial differential equations, Finite difference schemes, Consistency, Gröbner basis, Involutive algorithm

1. INTRODUCTION

Since, apart from very special cases, partial differential equations (PDEs) can only be solved numerically, the construction of their numerical solutions is a fundamental task in science and engineering. Among three classical numerical methods that are widely used for numerical solving of

PDEs the finite difference method¹ is the oldest one and is based upon the application of a local Taylor expansion to approximate the differential equations by difference ones [1, 2] defined on the chosen computational grid. The difference equations that approximate differential equations in the system of PDEs form its finite difference approximation (FDA) which together with discrete approximation of initial or/and boundary conditions is called finite-difference scheme (FDS).

A good FDA has to mimic or inherit the algebraic structure of a differential system. In particular it has to reproduce such fundamental properties of the continuous equations as symmetries and conservation laws [3, 4]. Provided with appropriate initial or/and boundary conditions in their discrete form, the main requirement to the FDS is its *convergence*. The last means that the numerical solution approaches to the true solution to the PDE system as the grid spacings go to zero. Further important properties of FDS are *consistency* and *stability*. The former means that the difference equations in FDA are reduced to the original PDEs when the grid spacings vanish,² whereas the latter means that the error in the solution remains bounded under small perturbation in the numerical data. Consistency is necessary for convergence. In accordance to the Lax-Richtmyer equivalence theorem [1, 2] proved first for (scalar) linear PDEs and extended to some nonlinear equations [5], a consistent FDA to a PDE with the well-posed initial value (Cauchy) problem converges if and only if it is stable. Thus, the consistency check is an important step in analysis of difference schemes.

In this paper for a FDA to a linear PDE system on uniform and orthogonal grids we suggest another concept of consistency called strong consistency (s-consistency) which means consistency of the set of all linear difference consequences of the FDA with the set of linear differential consequences of the PDE system. This concept improves the concept of equation-wise consistency (e-consistency) of a FDA with a PDE system and also admits an algorithmic check. This check is done via construction of a Gröbner basis for the difference ideal generated by the FDA to linear differential polynomials in the PDE system. We show that every s-consistent FDA is e-consistent and the converse is not true. It means that an s-consistent FDA reproduces at the discrete level more algebraic properties of the PDE system than one which is e-consistent and s-inconsistent. For the algorithmic check of s-consistency we use the involutive algorithm [6, 7] which apart from the construction of a Gröbner basis allows

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

¹The other two methods are the finite element method and the finite volume method.

²In Section 3 we give a more precise definition of consistency.

also to verify easily well-posedness of the initial value problem for an analytic system of PDEs [8, 9] as a prerequisite of convergence for its FDS.

The structure of the paper is as follows. In Sect. 2 we shortly describe the mathematical objects with which we deal in the paper. In Sect. 3 for the uniform and orthogonal grids with equally spaced nodes we define s-consistency of a FDA to a system of PDEs and relate it with the underlying consistency properties of a difference Gröbner basis of the ideal generated by the polynomials in the FDA. The algorithmic verification of s-consistency is presented in Sect. 4. Then we illustrate the concepts and methods of the paper by some examples (Sect. 5). In Sect. 6 we consider peculiarities of consistency for the grids with different spacings and conclude in Sect. 7.

2. PRELIMINARIES

Let $\mathbf{x} := \{x_1, \dots, x_n\}$ be the set of n real (independent) variables and $\mathbb{K} := \mathbb{Q}(\mathbf{x})$ be the field of rational functions with rational coefficients. $\mathbb{K}$ is both a differential and a difference field [10], respectively, for the set $\{\partial_1, \dots, \partial_n\}$ of derivation operators and the set $\{\sigma_1, \dots, \sigma_n\}$ of the differences acting on the functions $\phi \in \mathbb{K}$ as the right-shift operators $\sigma_i \phi(x_1, \dots, x_n) = \phi(x_1, \dots, x_i + h_i, \dots, x_n)$. Here the shift parameters h_i can take positive real values.

We shall use the same notation $\mathbb{K}[u^{(1)}, \dots, u^{(m)}]$ for both differential and difference polynomial rings over $\mathbb{K}$ and denote them by $\mathcal{R}$ resp. $\tilde{\mathcal{R}}$. The differential (resp. difference) indeterminates $u^{(1)}, \dots, u^{(m)}$ will be considered for differential (resp. difference) equations as dependent variables, and sometimes we shall use also the vector notation $\mathbf{u} := (u^{(1)}, \dots, u^{(m)})$. The subset of the differential ring $\mathcal{R}$ containing linear polynomials will be denoted by $\mathcal{R}_L$, and the linear subset of the difference ring by $\tilde{\mathcal{R}}_L$.

Hereafter we consider PDE systems of the form

$$f_1 = \dots = f_p = 0, \quad F := \{f_1, \dots, f_p\} \subset \mathcal{R}_L. \quad (1)$$

To approximate the differential system (1) by a difference one we shall use an orthogonal and uniform computational grid (mesh) as the set of points $(k_1 h_1, \dots, k_n h_n)$ in $\mathbb{R}^n$. Here $\mathbf{h} := (h_1, \dots, h_n)$ ($h_i > 0$) is the tuple of mesh steps (grid spacings) and the integer-valued vector $\mathbf{k} := (k_1, \dots, k_n) \in \mathbb{Z}^n$ numerates the grid points. If the actual solution to the problem (1) is given by $\mathbf{u}(\mathbf{x})$ then its approximation in the grid node will be given by the grid (vector) function $\mathbf{u}_{k_1, \dots, k_n} = \mathbf{u}(k_1 h_1, \dots, k_n h_n)$.

In the finite difference method derivatives in (1) are approximated by finite differences. This can be done in many ways. For example, the first-order derivative can be approximated by the forward difference

$$\partial_{x_j} u^{(i)} = \Delta_j(u^{(i)}) + O(h_j),$$

where

$$\Delta_j(u^{(i)}) := \frac{u_{k_1, \dots, k_j+1, \dots, k_n}^{(i)} - u_{k_1, \dots, k_j, \dots, k_n}^{(i)}}{h_j} \quad (2)$$

or by the centered difference

$$\partial_{x_j} u^{(i)} = \frac{u_{k_1, \dots, k_j+1, \dots, k_n}^{(i)} - u_{k_1, \dots, k_j-1, \dots, k_n}^{(i)}}{2h_j} + O(h_j^2). \quad (3)$$

By substituting finite differences for derivatives into system (1) and applying appropriate right-shift operators from

the monoid generated by $\{\sigma_1, \dots, \sigma_n\}$ to remove negative shifts in indices which may come out of expressions like (3) we obtain a FDA to (1) of the form

$$\tilde{f}_1 = \dots = \tilde{f}_p = 0, \quad \tilde{F} := \{\tilde{f}_1, \dots, \tilde{f}_p\} \subset \tilde{\mathcal{R}}_L. \quad (4)$$

In [11] another approach to generation of FDA was suggested. It is based on the finite volume method and on difference elimination. That approach is algorithmic and for nonlinear equations it can construct FDAs that cannot be obtained by the straightforward substitution of finite differences for derivatives into the differential equations. An example of such approximation was constructed in [11] for the Falkovich-Karman differential equation describing transonic flow in gas dynamics. Whereas the underlying differential equation is quadratically nonlinear, the obtained difference approximation is cubically nonlinear. Due to this fact the corresponding FDS reveals better numerical behavior than known quadratically nonlinear schemes.

3. CONSISTENCY

Here and in the next two sections we consider orthogonal and uniform grids with equisized mesh steps $h_1 = \dots = h_n = h$. First, we give the generally accepted definition [1, 2] of consistency of a single differential equation with its difference approximation.

Definition 1. Given a PDE $f = 0$ and a FDA $\tilde{f} = 0$, the FDA is said to be *consistent* with the PDE if for any smooth, i.e. sufficiently differentiable for the context, vector-function $\mathbf{u}(\mathbf{x})$

$$f(\mathbf{u}) - \tilde{f}(\mathbf{u}) \rightarrow 0 \quad \text{as } h \rightarrow 0,$$

the convergence being pointwise at each point $\mathbf{x}$.

Definition 1 allows to verify easily the consistency of $\tilde{f}$ with f by using the Taylor expansion of $\tilde{f}$ about a grid point which is non-singular for its coefficients. As a simple example consider the advection (or one-way wave) equation

$$f(u) = 0, \quad f(u) := u_x + \nu u_y \quad (\nu = \text{const}), \quad (5)$$

which is the simplest hyperbolic PDE. Its discretization by using the forward differences (2) for the derivatives gives

$$\tilde{f}(u) := \frac{u_{i+1,j} - u_{i,j}}{h} + \nu \frac{u_{i,j+1} - u_{i,j}}{h}. \quad (6)$$

The Taylor expansion about the grid point $(x = ih, y = jh)$ yields

$$\begin{aligned} u_{i+1,j} &= u_{i,j} + h u_x + \frac{h^2}{2} u_{xx} + O(h^3), \\ u_{i,j+1} &= u_{i,j} + h u_y + \frac{h^2}{2} u_{yy} + O(h^3), \end{aligned}$$

and thus

$$f(u) - \tilde{f}(u) = -\frac{h}{2} (u_{xx} + \nu u_{yy}) + O(h^2) \xrightarrow{h \rightarrow 0} 0.$$

This shows the consistency of (6) with (5).

If one considers a system of PDEs and performs its equation-wise discretization, as it is usually done in practice, then a natural generalization of Definition 1 to systems of equations is as follows.

Definition 2. Given a PDE system (1) and its difference approximation (4), we shall say that (4) is *equation-wise consistent* or *e-consistent* with (1) if every difference equation in (4) is consistent with the corresponding differential equation in (1).

In fact, in the literature only e-consistency of FDA to systems of PDEs is considered. However e-consistency may not be satisfactory in view of inheritance of properties of the differential systems at the discrete level.

We are now going to introduce another concept of consistency for difference approximations to PDE systems which strengthens Definition 2 and provides consistency of the (infinite) subset of $\tilde{\mathcal{R}}_L$ of all linear difference consequences of the discrete system (4) with the subset of $\mathcal{R}_L$ of all linear differential consequences of the PDE system (1).

To formulate the new concept we need the following definition.

Definition 3. We shall say that a *difference equation* $\tilde{f}(\mathbf{u}) = 0$ implies the *differential equation* $f(\mathbf{u}) = 0$ and write $\tilde{f} \triangleright f$ when the Taylor expansion about a grid point yields

$$\tilde{f}(\mathbf{u}) \xrightarrow{h \rightarrow 0} f(\mathbf{u})h^k + O(h^{k+1}), \quad k \in \mathbb{Z}_{\geq 0}. \quad (7)$$

It is clear that in this terminology, Definition 1 means $\tilde{f} \triangleright f$. Now we give our main definition.

Definition 4. Given a PDE system (1) and its difference approximation (4), we shall say that (4) is *strongly consistent* or *s-consistent* with (1) if

$$\forall \tilde{f} \in \langle \tilde{F} \rangle \cap \tilde{\mathcal{R}}_L \exists f \in \langle F \rangle \cap \mathcal{R}_L : \tilde{f} \triangleright f. \quad (8)$$

Comparing Definitions 2 and 4 one sees that s-consistency implies e-consistency. The converse is not true as shown by explicit examples in Sect. 5.

The s-consistency admits an algorithmic verification which is based on the following statement.

THEOREM 1. A difference approximation (4) to a differential system (1) is s-consistent if and only if any reduced Gröbner basis $\tilde{G} \subset \tilde{\mathcal{R}}_L$ of the difference ideal $\langle \tilde{F} \rangle$ satisfies

$$\forall \tilde{g} \in \tilde{G} \exists g \in \langle F \rangle \cap \mathcal{R}_L : \tilde{g} \triangleright g. \quad (9)$$

PROOF. Let $\succ$ be a difference ranking [10] and $\tilde{G}$ be a reduced difference Gröbner basis [10, 12] of $\langle \tilde{F} \rangle$ for this ranking satisfying the condition (9). Denote by G the set of differential polynomials that are implied by the elements in $\tilde{G}$. Consider a linear difference polynomial $\tilde{f} \in \langle \tilde{F} \rangle \cap \tilde{\mathcal{R}}_L$ and its standard representation (cf. [13]) w.r.t. $\tilde{G}$ and $\succ$ as a finite sum of the form

$$\begin{cases} \tilde{f} = \sum_{\tilde{g} \in \tilde{G}} \sum_{\mu} a_{\mu} \sigma^{\mu} \circ \tilde{g}, & a_{\mu} \in \mathbb{K}, \\ \forall \tilde{g}, \mu : \sigma^{\mu} \circ \text{ld}(\tilde{g}) \preceq \text{ld}(\tilde{f}). \end{cases} \quad (10)$$

Here $\text{ld}(q)$ denotes the *leader* [10] of a difference polynomial q , and we use the multiindex notation

$$\mu := (\mu_1, \dots, \mu_n) \in \mathbb{Z}_{\geq 0}^n, \quad \sigma^{\mu} := \sigma_1^{\mu_1} \circ \dots \circ \sigma_n^{\mu_n}.$$

Choose now a grid point, nonsingular for the sum in (10), and consider its Taylor expansion (in grid spacing h) about this point. The shift operators σ_j ($j = 1, \dots, n$) which occur in σ^{μ} and in $\tilde{g} \in \tilde{G}$ are expanded in the Taylor series

$$\sigma_j = \sum_{k \geq 0} h^k \partial_j^k \quad (11)$$

along with the shifted rational functions in the independent variables.

The representation (10) guarantees that the highest ranking partial derivatives which occur in the leading order in h and come from different elements of the Gröbner basis cannot cancel. Thereby, due to the condition (9), in the leading order in h , the Taylor expansion of $\tilde{f}$ will contain a finite sum of the form

$$f := \sum_{g \in G} \sum_{\mu} b_{\mu} \sigma^{\mu} \circ g, \quad b_{\mu} \in \mathbb{K}, \quad (12)$$

and hence $\tilde{f} \triangleright f \in \langle F \rangle \cap \mathcal{R}_L$.

Since $\tilde{G} \subset \langle \tilde{F} \rangle \cap \tilde{\mathcal{R}}_L$, the converse is trivially true. $\square$

COROLLARY 1. Let a FDA $\tilde{F} \subset \tilde{\mathcal{R}}_L$ be s-consistent with a set $F \subset \mathcal{R}_L$, then

$$\forall \tilde{p} \in \langle \tilde{F} \rangle \exists p \in \langle F \rangle : \tilde{p} \triangleright p. \quad (13)$$

PROOF. Consider a difference polynomial $\tilde{q} \in \tilde{\mathcal{R}}$ as a grid function. If one applies the Taylor expansion (11) of the shift operators about a grid point, then in the limit $h \rightarrow 0$ this polynomial takes the form

$$\tilde{q} = h^k q + O(h^{k+1}), \quad k \in \mathbb{Z}_{\geq 0},$$

where $q \in \mathcal{R}$ is a differential polynomial.

If now we multiply both sides of the representation (10) by a polynomial $\tilde{q}$, apply a finite number of the shift operators σ_i to the product and apply the Taylor expansion about a grid point to the result, then in the leading order in h we obtain the differential polynomial which results from the linear differential polynomial of the form (12) by its multiplication by q and applying finitely many derivations ∂_j .

Clearly, before doing the Taylor expansion one can also multiply the r.h.s. in (10) and apply the shift operations to the product several times. Afterwards, the leading (in h) order of the expansion will yield the differential polynomial generated by elements in the differential polynomial set G that is implied by the Gröbner basis of $\langle \tilde{F} \rangle$. $\square$

If one uses a minimal difference involutive basis [7, 9], then the representation (10) is unique with operators σ^{μ} being products of multiplicative differences.

It should be also noted that the condition (8) does not exploit the equality $\text{card } F = \text{card } \tilde{F}$ of the cardinalities for sets of differential and difference equations as is assumed in Definition 2. The equality of cardinalities is not used in the proof of Theorem 1 either. Therefore, both Definition 4 and Theorem 1 are relevant to the case when the FDA has different number of equations than the PDE system.

4. ALGORITHMIC CHECK

Given a finite set $F \subset \mathcal{R}_L$ of linear differential polynomials and its FDA $\tilde{F} \subset \tilde{\mathcal{R}}_L$, one can algorithmically verify whether $\tilde{F}$ is s-consistent with F . For a difference polynomial $\tilde{f} \in \tilde{\mathcal{R}}_L$ its consistency (e-consistency) with a differential polynomial $f \in \mathcal{R}_L$, i.e. condition $\tilde{f} \triangleright f$, can be algorithmically verified by performing the Taylor expansion of f in the grid spacing h . The condition $g \in \langle F \rangle \cap \mathcal{R}_L$ can also be algorithmically verified by construction of a Gröbner basis of the differential ideal $\langle F \rangle$.

The following algorithm verifies s-consistency of a finite set $\tilde{F} \subset \tilde{\mathcal{R}}_L$ of linear difference polynomials as FDA to a finite set $F \subset \mathcal{R}_L$ of linear partial differential polynomials. The algorithm uses Janet bases [7, 8] for both differential

and difference ideals, though reduced Gröbner bases or other involutive bases can also be used.

Algorithm: ConsistencyCheck ($F, \tilde{F}$)

```

1: choose differential resp. difference ranking  $\succ_1, \succ_2$ 
2:  $J := \text{JanetBasis}(F, \succ_1)$ 
3:  $\tilde{J} := \text{JanetBasis}(\tilde{F}, \succ_2)$ 
4:  $S := \text{true}$ 
5: while  $\tilde{J} \neq \emptyset$  and  $S = \text{true}$  do
6:   choose  $\tilde{g} \in \tilde{J}$ 
7:    $\tilde{J} := \tilde{J} \setminus \{\tilde{g}\}$ 
8:   compute  $g$  such that  $\tilde{g} \triangleright g$ 
9:   if  $\text{NF}_{\mathcal{J}}(g, J) \neq 0$  then
10:     $S := \text{false}$ 
11:   fi
12: od
13: return  $S$ 

```

The subalgorithm **JanetBasis** invoked in lines 2 and 3 computes the differential and difference Janet basis, respectively. The subalgorithm $\text{NF}_{\mathcal{J}}$ on line 9 computes the differential involutive normal form [8] of a linear differential polynomial g modulo J , and thereby checks whether $g \in \langle J \rangle \cap \mathcal{R}_L$. The subscript $\mathcal{J}$ indicates that the normal form is computed for Janet division.

Correctness of the algorithm **ConsistencyCheck** follows from Theorem 1 and from the fact that the Janet bases are Gröbner ones. Its *termination* is an obvious consequence of the finiteness of the set $\tilde{J}$, termination of the subalgorithms and the Taylor expansion step of line 8.

Algorithm: JanetBasis ($F, \succ$)

Input: $F \subset \mathcal{R}_L$ (resp. $\tilde{\mathcal{R}}_L$), a finite set; $\succ$, a ranking
Output: J , a Janet basis of $\langle F \rangle$

```

1: choose  $f \in F$  with lowest  $\text{ld}(f)$  w.r.t.  $\succ$ 
2:  $J := \{f\}$ ;  $Q := F \setminus \{f\}$ 
3: while  $Q \neq \emptyset$  do
4:    $h := 0$ 
5:   while  $Q \neq \emptyset$  and  $h = 0$  do
6:     choose  $q \in Q$  with lowest  $\text{ld}(q)$  w.r.t.  $\succ$ 
7:      $Q := Q \setminus \{q\}$ ;  $h := \text{NF}_{\mathcal{J}}(q, J)$ 
8:   od
9:   if  $h \neq 0$  then
10:    for all  $\{g \in J \mid \text{ld}(g) \succ \text{ld}(h)\}$  do
11:       $J := J \setminus \{g\}$ 
12:       $Q := Q \cup \{g\} \setminus \{\vartheta \circ g \mid \vartheta \in \text{NM}_{\mathcal{J}}(g, J)\}$ 
13:    od
14:     $J := J \cup \{h\}$ ;  $Q := Q \cup \{\vartheta \circ h \mid \vartheta \in \text{NM}_{\mathcal{J}}(h, J)\}$ 
15:   fi
16: od
17: return  $J$ 

```

For completeness of this paper we present also the **JanetBasis** algorithm in its simplest form. The algorithm computes the minimal Janet basis for both differential and difference ideals generated by the input set. The operator ϑ in lines 12 and 14 is either derivation or difference, and the set $\text{NM}_{\mathcal{J}}$ contains the Janet nonmultiplicative derivations (differences) for the polynomial g (line 12) and h (line 14). In its improved version this algorithm allows to compute

the reduced Gröbner basis in the course of the Janet basis computation, that is, without performing extra reductions to produce the former from the latter.

The algorithm **JanetBasis** has been implemented in Maple for differential and difference ideals in the form of the packages called Janet [14] and LDA (Linear Difference Algebra) [15]. Besides the main procedure, which computes involutive bases w.r.t. Janet or Janet-like division [16], commands that return the normal form of a linear differential or difference polynomial modulo an ideal and many tools for dealing with linear differential or difference operators are included; syzygies, Hilbert polynomials and series can be computed, and the set of standard monomials modulo an ideal (together with a Stanley decomposition) can be determined.

5. EXAMPLES

In this section we demonstrate the notion of strong consistency on some examples. The computations were carried out in a few seconds with the packages Janet and LDA in Maple 13 on an AMD Opteron machine. Alternatively, the Gröbner package in Maple in connection with the Ore algebra package [12] can be used to get the same results.

In the below examples difference approximations to the initial PDE systems are e-consistent by construction. We show, however, that *s-consistency* does not always hold for those approximations.

Example 1. Consider the overdetermined linear PDE system

$$u_x + yu_z + u = 0, \quad u_y + xu_w = 0 \quad (14)$$

for one unknown function u of four independent variables x, y, z, w . The minimal Janet basis J for the differential ideal in $\mathcal{R}$ generated by the left hand sides of (14) w.r.t. the degrevlex ranking with

$$\partial_x \succ \partial_y \succ \partial_z \succ \partial_w \quad (15)$$

contains an additional integrability condition and is completely given by

$$u_x + yu_w + u, \quad u_y + xu_w, \quad u_z - u_w. \quad (16)$$

It coincides with the reduced Gröbner basis for this ideal. First we choose forward differences (2) to discretize the original PDEs (14)

$$\Delta_1(u) + jh\Delta_3(u) + u_{i,j,k,l} = 0, \quad \Delta_2(u) + ih\Delta_4(u) = 0$$

at the grid point $x = ih, y = jh, z = kh, w = lh$. The minimal Janet basis $\tilde{J}_1$ (w.r.t. degrevlex with $\sigma_1 \succ \sigma_2 \succ \sigma_3 \succ \sigma_4$) for the difference ideal generated by these two linear difference polynomials $\tilde{f}_1, \tilde{f}_2$ coincides with the reduced Gröbner basis and consists of these polynomials (with leading terms $u_{i+1,j,k,l}$ resp. $u_{i,j+1,k,l}$) and three additional elements with leading terms $u_{i,j,k,l+2}, u_{i,j,k+1,l+1}, u_{i,j,k+2,l}$. For every difference polynomial $\tilde{f} \in \tilde{J}_1$ there exists $f \in \langle J \rangle \cap \mathcal{R}_L$ such that $\tilde{f} \triangleright f$, as can be checked by applying reduction modulo J to the Taylor expansion of $\tilde{f}$ about a grid point. Moreover, the set $\langle J \rangle \cap \mathcal{R}_L$ of differential polynomials implied by $\tilde{J}_1$ contains, in addition to equations (14), $yu_z - yu_w, u_z - u_w$ and $xu_z - xu_w$ which also show that the integrability condition $u_z - u_w$ is recovered as limit for $h \rightarrow 0$ from the discretization.

The discretization $\Delta_3(u) - \Delta_4(u)$ of $u_z - u_w$ has non-zero normal form modulo $\tilde{J}_1$. We add this difference polynomial

as another generator for the difference ideal in $\tilde{\mathcal{R}}$. The minimal Janet basis $\tilde{J}_2$ for this larger ideal is given by

$$\Delta_1(u) + u_{i,j,k,l}, \quad \Delta_2(u), \quad \Delta_3(u), \quad \Delta_4(u). \quad (17)$$

Now it is easy to check that the chosen discretization of (16) using forward differences is not s-consistent. We tried also some other discretizations of the differential Janet basis (16) and all of them were *s-inconsistent*. We conclude that it may be a non-trivial task to find a difference approximation of a Gröbner basis for an overdetermined set of partial differential polynomials that is strongly consistent.

Finally, we mention that the minimal Janet basis $\tilde{J}_3$ for the difference ideal generated by f_1, f_2 w.r.t. the elimination ranking with $\sigma_1 \succ \sigma_2 \succ \sigma_3 \succ \sigma_4$ contains the difference polynomial $\Delta_4^2 - ih^2 \Delta_3^2$ whose limit u_{ww} for $h \rightarrow 0$ is not an element of $\langle J \rangle \cap \mathcal{R}_L$. Moreover, if we add $\Delta_3(u) - \Delta_4(u)$ as another generator as above, the minimal Janet basis w.r.t. this elimination ranking equals (17).

Example 2. Consider the linear PDE system of two equations

$$u_{xxy} + v_x = 0, \quad u_{xyy} + v_y = 0 \quad (18)$$

for two unknown functions $u^{(1)} = u, u^{(2)} = v$ of two independent variables x, y . The left hand sides in (18) form a minimal Janet basis J (and reduced Gröbner basis) w.r.t. the ranking (15) for the ideal they generate. Using forward differences first to discretize (18) we get

$$\Delta_1^2 \Delta_2(u) + \Delta_1(v) = 0, \quad \Delta_1 \Delta_2^2(u) + \Delta_2(v) = 0. \quad (19)$$

The left hand sides form a Gröbner basis for the difference ideal in $\tilde{\mathcal{R}}$ they generate. It is easily verified by the consistency check (Sect. 4) that (19) is s-consistent with (18).

We now modify the discretization (19) slightly by using two-step forward differences

$$\Delta_{2,1}(v) := \frac{v_{i+2,j} - v_{i,j}}{2h}, \quad \Delta_{2,2}(v) := \frac{v_{i,j+2} - v_{i,j}}{2h},$$

i.e. the centered difference (3) w.r.t. the point $(x = (i+1)h, y = (j+1)h)$ instead of the one-step forward differences (2) for the second summands in (19). Thus, we consider

$$\Delta_1^2 \Delta_2(u) + \Delta_{2,1}(v) = 0, \quad \Delta_1 \Delta_2^2(u) + \Delta_{2,2}(v) = 0. \quad (20)$$

In this case, the left hand sides D_1, D_2 in (20) do not form a Gröbner basis for the ideal they generate, but the non-zero polynomial

$$\Delta_2(D_1) - \Delta_1(D_2) = (\Delta_2 \Delta_{2,1} - \Delta_1 \Delta_{2,2})(v)$$

has to be included as well. The Taylor expansion of this difference polynomial about a grid point has limit $v_{xyy} - v_{xxy}$ for $h \rightarrow 0$, which is not an element of $\langle J \rangle \cap \mathcal{R}_L$. Hence, the difference approximation (20) is not s-consistent with (18).

However, the following three FDA are strongly consistent with (18): two-step forward difference for ∂_x and one-step forward difference for ∂_y :

$$\Delta_{2,1}^2 \Delta_2(u) + \Delta_{2,1}(v), \quad \Delta_{2,1} \Delta_2^2(u) + \Delta_2(v);$$

shifted centered difference for ∂_x (i.e. $\sigma_1(\sigma_1 - \sigma_1^{-1})/(2h)$) and forward difference for ∂_y :

$$\Delta_{2,1}^2 \Delta_2(u) + \sigma_1 \Delta_{2,1}(v), \quad \Delta_{2,1} \Delta_2^2(u) + \sigma_1 \Delta_2(v);$$

shifted centered differences for both ∂_x and ∂_y :

$$\Delta_{2,1}^2 \Delta_{2,2}(u) + \sigma_1 \sigma_2 \Delta_{2,1}(v), \quad \Delta_{2,1} \Delta_{2,2}^2(u) + \sigma_1 \sigma_2 \Delta_{2,2}(v).$$

These three difference systems form reduced Gröbner bases for the difference ideals they generate, and the consistency check gives an affirmative answer in each case.

Example 3. The linear PDE system

$$f_1 := u_{xz} + yu = 0, \quad f_2 := u_{yw} + zu = 0 \quad (21)$$

for one unknown function u of four independent variables x, y, z, w has minimal Janet basis w.r.t. the ranking (15)

$$yu_y - zu_z, \quad u_x - u_w, \quad u_{zw} + yu.$$

We have the following two integrability conditions (see [9]) for f_1, f_2 :

$$\begin{aligned} &(\partial_{yy} u_{yw} + 2z \partial_{yw} + z^2) f_1 \\ &\quad - (\partial_{xyz} + z \partial_{xz} + y \partial_{yw} - \partial_x + 2 \partial_w + yz) f_2 = 0, \\ &(\partial_{xy} u_{zw} + z \partial_{xz} + y \partial_{yw} + 2 \partial_x - \partial_w + yz) f_1 \\ &\quad - (\partial_{xxz} + 2y \partial_{xz} + y^2) f_2 = 0. \end{aligned}$$

They form a reduced Gröbner basis for the ideal of all linear partial differential relations satisfied by f_1, f_2 , as can be checked by a syzygy computation with the Janet package. A more compact way to write these integrability conditions is as follows:

$$\begin{aligned} &((\partial_x \partial_z + y)(\partial_y \partial_w + z) - \partial_w + \partial_x) f_1 - (\partial_x \partial_z + y)^2 f_2 = 0, \\ &(\partial_y \partial_w + z)^2 f_1 - ((\partial_y \partial_w + z)(\partial_x \partial_z + y) + \partial_w - \partial_x) f_2 = 0. \end{aligned}$$

First we use forward differences (2) to discretize (21) at the grid point $x = ih, y = jh, z = kh, w = lh$:

$$\tilde{f}_1 := (\Delta_1 \Delta_3)(u) + jhu_{i,j,k,l}, \quad \tilde{f}_2 := (\Delta_2 \Delta_4)(u) + khu_{i,j,k,l}.$$

The minimal Janet basis (and reduced Gröbner basis) w.r.t. degrevlex (with $\sigma_1 \succ \sigma_2 \succ \sigma_3 \succ \sigma_4$) for the difference ideal generated by $\tilde{f}_1$ and $\tilde{f}_2$ is

$$\Delta_1(u) - jh^2 u_{i,j,k,l}, \quad u_{i,j+1,k,l}, \quad u_{i,j,k+1,l}, \quad \Delta_4(u) - kh^2 u_{i,j,k,l}.$$

It is easily verified using the consistency check of Sect. 4 that the FDA $\tilde{f}_1, \tilde{f}_2$ is not s-consistent.

Let us exchange $f_1 = 0$ in (21) by another linear PDE: $f_3 := u_{xy} + zu = 0$. It is a consequence of (21):

$$f_3 = -(\partial_y^2 \partial_w + z \partial_y) f_1 + (\partial_x \partial_y \partial_z + y \partial_y + 2) f_2.$$

However, the PDE system

$$f_2 = 0, \quad f_3 = 0 \quad (22)$$

is not equivalent to (21). It admits the following strongly consistent FDA:

$$\tilde{f}_2 := (\Delta_2 \Delta_4)(u) + khu_{i,j,k,l}, \quad \tilde{f}_3 := (\Delta_1 \Delta_2)(u) + khu_{i,j,k,l}.$$

In fact, the minimal Janet basis for (22) is $\{u_x - u_w, u_{yw} + zu\}$, and the reduced Gröbner basis for the difference ideal generated by $\tilde{f}_2, \tilde{f}_3$ is

$$(\Delta_1 - \Delta_4)(u), \quad (\Delta_2 \Delta_4)(u) + khu_{i,j,k,l},$$

which is easily checked to be s-consistent with (22). We note that if we discretize the integrability condition

$$(\partial_{xy} + z) f_2 - (\partial_{yw} + z) f_3 = 0 \quad (23)$$

for (22) with forward differences, we get

$$(\Delta_1 \Delta_2 + kh) \tilde{f}_2 - (\Delta_2 \Delta_4 + kh) \tilde{f}_3 = 0,$$

i.e. the discretization of (23) is satisfied.

In contrast to the previous PDE system we consider now

$$f_1 = 0, \quad f_3 = 0. \quad (24)$$

It is not equivalent to (21) either. In this case, if we discretize with forward differences,

$$\tilde{f}_1 := (\Delta_1 \Delta_3)(u) + jhu_{i,j,k,l}, \quad \tilde{f}_3 := (\Delta_1 \Delta_2)(u) + kh u_{i,j,k,l},$$

we obtain an FDA which is not s-consistent with (24). In fact, the minimal Janet basis for the difference ideal is $\{u\}$ having only the zero solution.

We could have predicted this collapse of solutions by examining the following integrability condition: $(\partial_{xy} + z)f_1 - (\partial_{xz} + y)f_3 = 0$.

We discretize it with forward differences:

$$\begin{aligned} & (\Delta_1 \Delta_2 + kh)\tilde{f}_1 - (\Delta_1 \Delta_3 + jh)\tilde{f}_3 \\ &= \Delta_1 \Delta_2(jhu) - jh\Delta_1 \Delta_2(u) + kh\Delta_1 \Delta_3(u) - \Delta_1 \Delta_3(khu) \\ &= h\Delta_1((k\Delta_3 - \Delta_3k)(u) - (j\Delta_2 - \Delta_2j)(u)) \\ &= \frac{1}{h}(u_{i+1,j+1,k,l} - u_{i,j+1,k,l} - u_{i+1,j,k+1,l} + u_{i,j,k+1,l}). \end{aligned}$$

This discretization has limit $u_{xz} - u_{xy}$ for $h \rightarrow 0$, whose normal form modulo the Janet basis for (24) is $(z - y)u$, i.e., $u = 0$ is implied.

One can check that the FDA $\{\tilde{f}_1, \tilde{f}_2, \tilde{f}_3\}$ is not s-consistent with

$$f_1 = 0, \quad f_2 = 0, \quad f_3 = 0; \quad (25)$$

the discretizations of the two integrability conditions of order four given in the beginning of this example have a non-zero limit for $h \rightarrow 0$ modulo the Janet basis for (25).

6. GRID WITH DIFFERENT SPACINGS

For an orthogonal and uniform grid with the spacings $\mathbf{h} := (h_1, \dots, h_n)$ Definition 1 of consistency for a FDA with a PDE can be reformulated as the condition

$$f(\mathbf{u}) - \tilde{f}(\mathbf{u}) \rightarrow 0 \quad \text{as } |\mathbf{h}| \rightarrow 0 \quad (i = 1, \dots, n), \quad (26)$$

where $|\mathbf{h}| \rightarrow 0$ means $h_1, \dots, h_n \rightarrow 0$.

In some cases, however, one has to restrict the manner in which $|\mathbf{h}| \rightarrow 0$. Consider again the advection equation (5) and its difference approximation in the Lax-Friedrichs form [2]

$$\tilde{f} = \frac{2u_{i+1,j+1} - u_{i,j+2} - u_{i,j}}{2h_1} + \nu \frac{u_{i,j+2} - u_{i,j}}{2h_2}. \quad (27)$$

The Taylor expansion of $\tilde{f}$ about the point $x = h_1 i, y = h_2(j+1)$ reads

$$\begin{aligned} \tilde{f} &= u_x + \nu u_y + \frac{h_1}{2} u_{xx} - \frac{h_2^2}{2h_1} u_{yy} + \nu \frac{h_2^2}{6} u_{yyy} + \frac{1}{6} \nu u_{xxx} h_1^2 \\ &+ \frac{1}{6} \nu u_{xxx} h_1^2 - \frac{h_2^4}{24h_1} u_{xxxx} + \nu \frac{h_2^4}{120} u_{xxxxx} + O(h_1^3 + \frac{h_2^6}{h_1} + h_2^6). \end{aligned}$$

It shows that the consistency with (5) holds only if $h_1 \rightarrow 0$ and $h_2^2/h_1 \rightarrow 0$ (cf. [2]).

Respectively, Definition 2 of e-consistency for systems of linear PDEs discretized on the general orthogonal and uniform grids has the following form.

Definition 5. A difference approximation (4) to (1) is *e-consistent* if there is a passage to the limit $|\mathbf{h}| \rightarrow 0$ which provides consistency of every difference equation in (4) with the corresponding differential equation in (1) by doing the Taylor expansion about a grid point.

The search for such a passage by analyzing the multivariate Taylor expansion of every equation in the difference system (4) generally can be problematic and computationally cumbersome. We shall not consider this problem and adopt Definition 3 to the grid under consideration.

Definition 6. A difference approximation to a PDE system is *s-consistent* with this system if there is a passage to the limit $|\mathbf{h}| \rightarrow 0$ such that the following holds:

$$\forall \tilde{f} \in \langle \tilde{F} \rangle \cap \tilde{\mathcal{R}}_L \exists f \in \langle F \rangle \cap \mathcal{R}_L : \tilde{f} \triangleright f. \quad (28)$$

Now instead of straightforward reformulation of Theorem 1 for the grid with different spacings we restate it as follows.

THEOREM 2. *A passage to the limit $|\mathbf{h}| \rightarrow 0$ providing the fulfillment of condition (28) exists if and only if there is a passage to the limit for a reduced Gröbner basis $\tilde{G} \subset \tilde{\mathcal{R}}_L$ of the difference ideal $\langle \tilde{F} \rangle$ such that*

$$\forall \tilde{g} \in \tilde{G} \exists g \in \langle F \rangle \cap \mathcal{R}_L : \tilde{g} \triangleright g,$$

and for every such passage the condition (28) is satisfied.

PROOF. It can be easily seen from the proof of Theorem 1 that the same reasoning is applicable in this case, too. $\square$

7. CONCLUSION

We have shown that for a uniform and orthogonal solution grid a Gröbner basis of the difference ideal generated by a discretized linear system of PDEs contains important information on quality of the discretization, namely, on consistency of its linear difference consequences with the linear consequences of the PDE system. This property that we call s(strong)-consistency is superior to the in practice commonly used concept of consistency of the difference equations with their differential counterparts.

Even rather simple examples in Sect. 5 demonstrate that for overdetermined systems of PDEs the problem of constructing their s-consistent discretization may be a nontrivial problem. The algorithmic consistency check (Sect. 4) does not give answer how to construct a strongly consistent FDA for such systems. The algorithmic approach to generation of FDA suggested in [11] provides a more regular procedure for constructing a good FDA, since it exploits the conservation law form of the PDE system, when it admits such form, and preserves this form at the discrete level. Since conservation laws, if they are not explicitly incorporated into the PDE system, can always be expressed (linearly in the case of linear PDEs) in terms of integrability conditions (cf. [9], ch.2), the completion of the system to involution (or construction of its differential Gröbner basis) is an important step of its preprocessing before numerical solving. It is well known that conservation laws need special care in numerical solving of PDEs [3]. Thus, the last equation in (16) being the integrability condition for system (14) has the conservation law form.

Our algorithmic check of s-consistency is based on completion to involution (or construction of a Gröbner basis which is a formally integrable PDEs system [9] in the differential case) for both differential and difference systems. In addition to the consistency verification, if the initial differential system of the form $F \subset \mathcal{R}_L$ is involutive for an orderly (Riquier) ranking, then it admits formal well-posing of the initial value problem in the domain where none of the leading coefficients

and none of the coefficient denominators vanish (cf. [8, 9, 17]). In view of the Lax-Richtmyer theorem [1, 2] this provides the necessary condition for convergence of a numerical solution to the exact one when the grid spacings go to zero. Another necessary condition for convergence is stability. For many discretizations the latter may hold only under certain restrictions on the grid spacings. For example, difference approximations (6) and (27) are stable only if $|\nu h_1/h_2| \leq 1$ (Courant-Friedrichs-Levy stability condition [1, 2]).

For grids with unequal spacings the consistency verification may be more difficult because of the restrictions on the passage to the limit in (26) and respectively in checking s-consistency conditions (28). However, such situation arises not very often in practice when any passage to zero in (26) (resp. in (28)) is acceptable.

Extension of the results in the paper to nonlinear PDEs has such a principle obstacle as nonexistence of Gröbner bases (except in very restricted cases) for differential ideals generated by nonlinear differential polynomials, cf. [18]. And even in the case of their existence their computation is only possible by hand since there is no software computing such Gröbner bases. Nevertheless, consideration of difference S -polynomials and the condition of their reducibility to zero modulo the set of polynomials in the difference approximation may be useful for verification of its consistency. This was demonstrated recently in [19] where the method of paper [11] was applied to the generation of FDA to two-dimensional Navier-Stokes equations, and for one of the constructed approximations its inconsistency was detected.

While nonlinear differential systems can be disjointly decomposed into algebraically simple and involutive subsystems [20], investigating whether nonlinear difference systems can be treated in a similar way is a new important research topic.

8. ACKNOWLEDGMENTS

The contribution of the first author (V.P.G.) was supported in part by grant 10-01-00200 from the Russian Foundation for Basic Research. The authors are grateful to Yuri Blinkov for helpful remarks.

9. REFERENCES

- [1] J.W.Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*, 2nd Edition. Springer-Verlag, New York, 1998.
- [2] J.C.Strikwerda. *Finite Difference Schemes and Partial Differential Equations*, 2nd Edition. SIAM, Philadelphia, 2004.
- [3] J.W.Thomas. *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations*. Springer-Verlag, New York, 1999.
- [4] V.Dorodnitsyn. *The group properties of difference equations*, Moscow, Fizmatlit, 2001 (in Russian).
- [5] E.E.Rosinger. *Nonlinear equivalence, reduction of PDEs to ODEs and fast convergent numerical methods*. Pitman, London, 1983.
- [6] V.P.Gerdt. *Involutive Algorithms for Computing Gröbner Bases*. Computational Commutative and Non-Commutative Algebraic Geometry. IOS Press, Amsterdam, 2005, pp. 199–225. arXiv:math.AC/0501111.
- [7] V.P.Gerdt. *Gröbner Bases Applied to Systems of Linear Difference Equations*. Physics of Particles and Nuclei Letters Vol.5, No.3, 2008, 425–436. arXiv:cs.SC/0611041
- [8] V.P.Gerdt. *Completion of Linear Differential Systems to Involution*, Computer Algebra in Scientific Computing / CASC 1999, Springer, Berlin, 1999, pp. 115–137. arXiv:math.AP/9909114.
- [9] W.M.Seiler. *Involution: The Formal Theory of Differential Equations and its Applications in Computer Algebra*. Algorithms and Computation in Mathematics 24, Springer, 2010.
- [10] A.Levin. *Difference Algebra*. Algebra and Applications 8, Springer, 2008.
- [11] V.P.Gerdt, Yu.A.Blinkov and V.V.Mozzhilkin. *Gröbner Bases and Generation of Difference Schemes for Partial Differential Equations*. Symmetry, Integrability and Geometry: Methods and Applications (SIGMA) 2, 051, 2006, 26 pages. arXiv:math.RA/0605334
- [12] F.Chyzak. *Gröbner Bases, Symbolic Summation and Symbolic Integration*. In: Gröbner Bases and Applications, B.Buchberger and F.Winkler (Eds.), Cambridge University Press, 1998.
- [13] T.Becker and V.Weispfenning. *Gröbner Bases: A Computational Approach to Commutative Algebra*. Graduate Texts in Mathematics 141, Springer, New York, 1993.
- [14] Yu.A.Blinkov, C.F.Cid, V.P.Gerdt, W.Plesken and D.Robertz. *The MAPLE Package Janet: I. Polynomial Systems. II. Linear Partial Differential Equations*. Proc. 6th Int.Workshop on Computer Algebra in Scientific Computing, Passau, 2003. Cf. also <http://wwwb.math.rwth-aachen.de/Janet>
- [15] V.P.Gerdt and D.Robertz. *A Maple Package for Computing Gröbner Bases for Linear Recurrence Relations*. Nuclear Instruments and Methods in Physics Research 559(1), 2006, 215–219. arXiv:cs.SC/0509070. Cf. also <http://wwwb.math.rwth-aachen.de/Janet>
- [16] V.P.Gerdt and Yu.A.Blinkov. *Janet-like Monomial Division*. Computer Algebra in Scientific Computing / CASC 2005, LNCS 3781, Springer-Verlag, Berlin, 2005, pp. 174–183; *Janet-like Gröbner Bases*, ibid., pp. 184–195.
- [17] A.G.Khovanskii and S.P.Chulkov. *Hilbert polynomial for a system of linear partial differential equations*. Research Report in Mathematics, No.4, Stockholm University, 2005.
- [18] A.Zobnin. *Admissible orderings and Finiteness Criteria for Differential Standard Bases*. Proceedings of ISSAC'2005, ACM Press, 2005, pp. 365–372.
- [19] V.P.Gerdt and Yu.A.Blinkov. *Involution and Difference Schemes for the Navier-Stokes Equations*. Computer Algebra in Scientific Computing / CASC 2009, LNCS 5743, Springer-Verlag, Berlin, 2009, pp. 94–105.
- [20] T.Bächler, V.P.Gerdt, M.Lange-Hegermann and D.Robertz. *Thomas Decomposition: I. Algebraic Systems; II. Differential Systems*. Submitted to Computer Algebra in Scientific Computing / CASC 2010 (September 5–12, 2010, Tsakhkadzor, Armenia).

Computation with Semialgebraic Sets Represented by Cylindrical Algebraic Formulas

Adam Strzeboński
Wolfram Research Inc.
100 Trade Centre Drive
Champaign, IL 61820, U.S.A.
adams@wolfram.com

ABSTRACT

Cylindrical algebraic formulas are an explicit representation of semialgebraic sets as finite unions of cylindrically arranged disjoint cells bounded by graphs of algebraic functions. We present a version of the Cylindrical Algebraic Decomposition (CAD) algorithm customized for efficient computation of arbitrary combinations of unions, intersections and complements of semialgebraic sets given in this representation. The algorithm can also be used to eliminate quantifiers from Boolean combinations of cylindrical algebraic formulas. We show application examples and an empirical comparison with direct CAD computation for unions and intersections of semialgebraic sets.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—*Algebraic algorithms*; G.4 [Mathematical Software]: Algorithm design and analysis

General Terms

Algorithms, Experimentation, Performance, Reliability

Keywords

semialgebraic sets, cylindrical algebraic decomposition, solving inequalities, quantifier elimination

1. INTRODUCTION

A *system of polynomial equations and inequalities* in variables $x_1, \dots, x_n$ is a formula

$$S(x_1, \dots, x_n) = \bigvee_{1 \leq i \leq l} \bigwedge_{1 \leq j \leq m} f_{i,j}(x_1, \dots, x_n) \rho_{i,j} 0$$

where $f_{i,j} \in \mathbb{R}[x_1, \dots, x_n]$, and each $\rho_{i,j}$ is one of $<, \leq, \geq, >, =$, or $\neq$.

A subset of $\mathbb{R}^n$ is *semialgebraic* if it is a solution set of a system of polynomial equations and inequalities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

A *quantified system of real polynomial equations and inequalities* in free variables $x_1, \dots, x_n$ and quantified variables $t_1, \dots, t_m$ is a formula

$$Q_1 t_1 \dots Q_m t_m S(t_1, \dots, t_m; x_1, \dots, x_n)$$

Where Q_i is $\exists$ or $\forall$, and S is a system of real polynomial equations and inequalities in $t_1, \dots, t_m, x_1, \dots, x_n$.

By Tarski's theorem (see [16]), solution sets of quantified systems of real polynomial equations and inequalities are semialgebraic.

Every semialgebraic set can be represented as a finite union of disjoint *cells* (see [12]), defined recursively as follows.

1. A cell in $\mathbb{R}$ is a point or an open interval.

2. A cell in $\mathbb{R}^{k+1}$ has one of the two forms

$$\begin{aligned} &\{(a_1, \dots, a_k, a_{k+1}) : (a_1, \dots, a_k) \in C_k \wedge \\ &\quad a_{k+1} = r(a_1, \dots, a_k)\} \\ &\{(a_1, \dots, a_k, a_{k+1}) : (a_1, \dots, a_k) \in C_k \wedge \\ &\quad r_1(a_1, \dots, a_k) < a_{k+1} < r_2(a_1, \dots, a_k)\} \end{aligned}$$

where C_k is a cell in $\mathbb{R}^k$, r is a continuous algebraic function, and r_1 and r_2 are continuous algebraic functions, $-\infty$, or ∞ , and $r_1 < r_2$ on C_k .

The Cylindrical Algebraic Decomposition (CAD) algorithm [5, 3, 15] can be used to compute a cell decomposition of any semialgebraic set presented by a quantified system of polynomial equations and inequalities. An alternative method of computing cell decompositions is given in [4]. Cell decompositions computed by the CAD algorithm can be represented directly [2, 15] as cylindrical algebraic formulas (CAF; a precise definition is given in the next section).

EXAMPLE 1. The following formula $F(x, y, z)$ is a CAF representation of a cell decomposition of the closed unit ball.

$$\begin{aligned} F(x, y, z) &:= x = -1 \wedge y = 0 \wedge z = 0 \vee \\ &\quad -1 < x < 1 \wedge b_2(x, y, z) \vee \\ &\quad x = 1 \wedge y = 0 \wedge z = 0 \\ b_2(x, y, z) &:= y = R_1(x) \wedge z = 0 \vee \\ &\quad R_1(x) < y < R_2(x) \wedge b_{2,2}(x, y, z) \vee \\ &\quad y = R_2(x) \wedge z = 0 \\ b_{2,2}(x, y, z) &:= z = R_3(x, y) \vee \\ &\quad R_3(x, y) < z < R_4(x, y) \vee \\ &\quad z = R_4(x, y) \end{aligned}$$

where

$$\begin{aligned} R_1(x) &= \text{Root}_{y,1}(x^2 + y^2) = -\sqrt{1-x^2} \\ R_2(x) &= \text{Root}_{y,2}(x^2 + y^2) = \sqrt{1-x^2} \\ R_3(x, y) &= \text{Root}_{z,1}(x^2 + y^2 + z^2) = -\sqrt{1-x^2-y^2} \\ R_4(x, y) &= \text{Root}_{z,2}(x^2 + y^2 + z^2) = \sqrt{1-x^2-y^2} \end{aligned}$$

The CAF representation of a semialgebraic set A can be used to decide whether A is nonempty, to find the minimal and maximal values of the first coordinate of elements of A , to generate an arbitrary element of A , to find a graphical representation of A , to compute the volume of A or to compute multidimensional integrals over A (see [14]).

A natural question to ask is how to compute set-theoretic operations with semialgebraic sets represented by cylindrical algebraic formulas. Set-theoretic operations on solution sets of formulas correspond to Boolean operations on the formulas, so the question is how to compute a cell decomposition of a semialgebraic set given by a Boolean combination of cylindrical algebraic formulas. In principle this could be done using an extension of the CAD algorithm to arbitrary systems of equations and inequalities involving algebraic functions. However, this is inefficient, as it requires introducing a new variable for each algebraic function that appears in the input. In this paper we present a customized version of the CAD algorithm which does not introduce new variables and makes efficient use of CAF structure, so that computation of cell decomposition for a semialgebraic set represented by a Boolean combination of CAFs is often faster than for the same set represented by a quantifier free polynomial system. The algorithm allows quantifiers, as long as the order of quantifier variables agrees with the order of variables in CAFs.

ALGORITHM 2. (CAFCombine)

Input: Cylindrical algebraic formulas

$$F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n)$$

a Boolean formula $\Phi(p_1, \dots, p_m)$ and a sequence of quantifiers $Q_1, \dots, Q_k$, with $0 \leq k \leq n$.

Output: A cylindrical algebraic formula $F(x_1, \dots, x_{n-k})$ equivalent to

$$Q_1 x_{n-k+1} \dots Q_k x_n \Phi(F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n))$$

EXAMPLE 3. (Solotareff problem for $n = 4$, [6])¹

Problem: Let

$$\begin{aligned} P_1(r, a, b, u, v) &:= (r^2 - 24r + 16 < 0 \vee r > 1) \wedge \\ &\quad -1 < u < v < 1 \wedge r - b > 0 \\ P_2(r, a, b, u, v) &:= v^3 + rv^2 - v^2 - av - rv + v - b + a + \\ &\quad r - 1 = 0 \wedge 4v^3 + 3rv^2 - 2av - b = 0 \\ P_3(r, a, b, u) &:= u^3 + ru^2 + u^2 - au + ru + u - b - a + \\ &\quad r + 1 = 0 \wedge 4u^3 + 3ru^2 - 2au - b = 0 \end{aligned}$$

Find a cell decomposition for the solution set of the formula

$$\exists u \in \mathbb{R} \exists v \in \mathbb{R} P_1(r, a, b, u, v) \wedge P_2(r, a, b, u, v) \wedge P_3(r, a, b, u) \quad (1.1)$$

Method 1: Use the CAD algorithm to find a CAF representing the solution set of (1.1).

¹We treat the system (1.1) as a benchmark CAD computation problem. For a more general solution of Solotareff's problem see [9].

Method 2: Use the CAD algorithm to find a polynomial system of equations and inequalities $P_4(r, a, b, u)$ equivalent to

$$\exists v \in \mathbb{R} P_1(r, a, b, u, v) \wedge P_2(r, a, b, u, v) \quad (1.2)$$

and then use the CAD algorithm to find a CAF representing the solution set of

$$\exists u \in \mathbb{R} P_3(r, a, b, u) \wedge P_4(r, a, b, u)$$

Method 3: Use the CAD algorithm to find CAFs $F_1(r, a, b, u)$ representing the solution set of (1.2) and $F_2(r, a, b, u)$ representing the solution set of $P_3(r, a, b, u)$ and then use CAFCombine to compute a CAF representing the solution set of

$$\exists u \in \mathbb{R} F_1(r, a, b, u) \wedge F_2(r, a, b, u)$$

The first two methods did not finish the computation in an hour. The third method solves the problem in the total time of 7.57 seconds.

EXAMPLE 4. (Unions and intersections of balls in $\mathbb{R}^3$)

Problem: Let

$$B_{v,k} := \{(x, y, z) \in \mathbb{R}^3 : (x - kv)^2 + (y - kv)^2 + (z - kv)^2 \leq 1\}$$

For given $v \in \mathbb{R}_+$ and $n \in \mathbb{N}_+$, find cell decompositions of

$$\begin{aligned} U_{v,n} &:= \bigcup_{k=0}^{n-1} B_{v,k} \\ I_{v,n} &:= \bigcap_{k=0}^{n-1} B_{v,k} \end{aligned}$$

Method 1: Use the CAD algorithm to find CAFs representing $U_{v,n}$ and $I_{v,n}$.

Method 2: Use the CAD algorithm to find CAFs representing $B_{v,k}$, for $0 \leq k \leq n-1$ and then use CAFCombine to compute CAFs representing $U_{v,n}$ and $I_{v,n}$.

Experiments suggest that for a fixed v as values of n grow the second method becomes significantly faster than the first method.

The last section of the paper contains more detailed experimental data.

2. CYLINDRICAL ALGEBRAIC FORMULAS

DEFINITION 5. A real algebraic function given by defining polynomial $f \in \mathbb{Z}[x_1, \dots, x_n, y]$ and root number $p \in \mathbb{N}_+$ is the function

$$\text{Root}_{y,p} f : \mathbb{R}^n \ni (x_1, \dots, x_n) \longrightarrow \text{Root}_{y,p} f(x_1, \dots, x_n) \in \mathbb{R} \quad (2.1)$$

where $\text{Root}_{y,p} f(x_1, \dots, x_n)$ is the p -th real root of f treated as a univariate polynomial in y . The function is defined for those values of $x_1, \dots, x_n$ for which $f(x_1, \dots, x_n, y)$ has at least p real roots. The real roots are ordered by the increasing value, counting multiplicities. A real algebraic number $\text{Root}_{y,p} f \in \mathbb{R}$ given by defining polynomial $f \in \mathbb{Z}[y]$ and root number p is the p -th real root of f . Let Alg be the set of real algebraic numbers and for $C \subseteq \mathbb{R}^n$ let Alg_C denote the set of all algebraic functions defined and continuous on C . (See [13, 14] for more details on how algebraic numbers and functions can be implemented in a computer algebra system.)

DEFINITION 6. A set $P \subseteq \mathbb{R}[x_1, \dots, x_n, y]$ is delineable over $C \subseteq \mathbb{R}^n$ iff

1. $\forall f \in P \exists k_f \in \mathbb{N} \forall a \in C \# \{b \in \mathbb{R} : f(a, b) = 0\} = k_f$.
2. For any $f \in P$ and $1 \leq p \leq k_f$, $\text{Root}_{y,p}f$ is a continuous function on C .
- 3.

$$\forall f, g \in P \quad (\exists a \in C \text{Root}_{y,p}f(a) = \text{Root}_{y,q}g(a) \Leftrightarrow \forall a \in C \text{Root}_{y,p}f(a) = \text{Root}_{y,q}g(a))$$

DEFINITION 7. A cylindrical system of algebraic constraints in variables $x_1, \dots, x_n$ is a sequence $A = (A_1, \dots, A_n)$ satisfying the following conditions.

1. For $1 \leq k \leq n$, A_k is a set of formulas

$$A_k = \{a_{i_1, \dots, i_k}(x_1, \dots, x_k) : 1 \leq i_1 \leq m \wedge 1 \leq i_2 \leq m_{i_1} \wedge \dots \wedge 1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}\}$$

2. For each $1 \leq i_1 \leq m$, $a_{i_1}(x_1)$ is true or

$$x_1 = r$$

where $r \in \text{Alg}$, or

$$r_1 < x_1 < r_2$$

where $r_1 \in \text{Alg} \cup \{-\infty\}$, $r_2 \in \text{Alg} \cup \{\infty\}$ and $r_1 < r_2$. Moreover, if $s_1, s_2 \in \text{Alg} \cup \{-\infty, \infty\}$, s_1 appears in $a_u(x_1)$, s_2 appears in $a_v(x_1)$ and $u < v$ then $s_1 \leq s_2$.

3. Let $k < n$, $I = (i_1, \dots, i_k)$ and let $C_I \subseteq \mathbb{R}^k$ be the solution set of

$$a_{i_1}(x_1) \wedge a_{i_1, i_2}(x_1, x_2) \wedge \dots \wedge a_{i_1, \dots, i_k}(x_1, \dots, x_k) \quad (2.2)$$

- (a) For each $1 \leq i_{k+1} \leq m_I$,

$$a_{i_1, \dots, i_k, i_{k+1}}(x_1, \dots, x_k, x_{k+1})$$

is true or

$$x_{k+1} = r(x_1, \dots, x_k) \quad (2.3)$$

and $r \in \text{Alg}_{C_I}$, or

$$r_1(x_1, \dots, x_k) < x_{k+1} < r_2(x_1, \dots, x_k) \quad (2.4)$$

where $r_1 \in \text{Alg}_{C_I} \cup \{-\infty\}$, $r_2 \in \text{Alg}_{C_I} \cup \{\infty\}$ and $r_1 < r_2$ on C_I .

- (b) If $s_1, s_2 \in \text{Alg}_{C_I} \cup \{-\infty, \infty\}$, s_1 appears in

$$a_{i_1, \dots, i_k, u}(x_1)$$

s_2 appears in

$$a_{i_1, \dots, i_k, v}(x_1)$$

and $u < v$ then $s_1 \leq s_2$ on C_I .

- (c) Let $P_I \subseteq \mathbb{Z}[x_1, \dots, x_k, x_{k+1}]$ be the set of defining polynomials of all real algebraic functions that appear in formulas a_J for $J = (i_1, \dots, i_k, i_{k+1})$, $1 \leq i_{k+1} \leq m_I$. Then P_I is delineable over C_I .

DEFINITION 8. Let A be a cylindrical system of algebraic constraints in variables $x_1, \dots, x_n$. Define

$$b_{i_1, \dots, i_n}(x_1, \dots, x_n) := \text{true}$$

For $2 \leq k \leq n$, level k cylindrical algebraic subformulas given by A are the formulas

$$b_{i_1, \dots, i_{k-1}}(x_1, \dots, x_n) := \bigvee_{1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}} a_{i_1, \dots, i_k}(x_1, \dots, x_k) \wedge b_{i_1, \dots, i_k}(x_1, \dots, x_n)$$

The support cell of $b_{i_1, \dots, i_{k-1}}$ is the solution set

$$C_{i_1, \dots, i_{k-1}} \subseteq \mathbb{R}^k$$

of

$$a_{i_1}(x_1) \wedge a_{i_1, i_2}(x_1, x_2) \wedge \dots \wedge a_{i_1, \dots, i_{k-1}}(x_1, \dots, x_{k-1})$$

The cylindrical algebraic formula (CAF) given by A is the formula

$$F(x_1, \dots, x_n) := \bigvee_{1 \leq i_1 \leq m} a_{i_1}(x_1) \wedge b_{i_1}(x_1, \dots, x_n)$$

REMARK 9. Let $F(x_1, \dots, x_n)$ be a CAF given by a cylindrical system of algebraic constraints A . Then

1. For $1 \leq k \leq n$, sets $C_{i_1, \dots, i_k}$ are cells in $\mathbb{R}^k$.

2. Cells

$$\{C_{i_1, \dots, i_n} : 1 \leq i_1 \leq m \wedge 1 \leq i_2 \leq m_{i_1} \wedge \dots \wedge 1 \leq i_n \leq m_{i_1, \dots, i_{n-1}}\}$$

form a decomposition of the solution set S_F of F , i.e. they are disjoint and their union is equal to S_F .

PROOF. Both parts of the remark follow from the definitions of A and F . $\square$

REMARK 10. Given a quantified system of polynomial equations and inequalities with free variables $x_1, \dots, x_n$ a version of the CAD algorithm can be used to find a CAF $F(x_1, \dots, x_n)$ equivalent to the system.

PROOF. The version of CAD described in [15] returns a CAF equivalent to the input system. $\square$

3. THE MAIN ALGORITHM

In this section we describe the algorithm *CAFCombine*. The algorithm is a modified version of the CAD algorithm. We describe only the modification. For details of the CAD algorithm see [3, 5]. Our implementation is based on the version of CAD described in [15].

DEFINITION 11. Let $P \subseteq \mathbb{R}[x_1, \dots, x_n]$ be a finite set of polynomials and let $\bar{P}$ be the set of irreducible factors of elements of P . $W = (W_1, \dots, W_n)$ is a projection sequence for P iff

1. Projection sets $W_1, \dots, W_n$ are finite sets of irreducible polynomials.
2. For $1 \leq k \leq n$, $\bar{P} \cap (\mathbb{R}[x_1, \dots, x_k] \setminus \mathbb{R}[x_1, \dots, x_{k-1}]) \subseteq W_k \subseteq \mathbb{R}[x_1, \dots, x_k] \setminus \mathbb{R}[x_1, \dots, x_{k-1}]$.
3. If $k < n$ and all polynomials of W_k have constant signs on a cell $C \subseteq \mathbb{R}^k$, then all polynomials of W_{k+1} that are not identically zero on $C \times \mathbb{R}$ are delineable over C .

REMARK 12. For an arbitrary finite set $P \subseteq \mathbb{R}[x_1, \dots, x_n]$ a projection sequence can be computed using Hong's projection operator [8]. McCallum's projection operator [10, 11] gives smaller projection sets for well-oriented sets P .

If $P \subseteq Q \subseteq \mathbb{R}[x_1, \dots, x_n]$ and W is a projection sequence for Q then W is a projection sequence for P .

NOTATION 13. For a CAF F , let P_F denote the set of defining polynomials of all algebraic numbers and functions that appear in F .

First let us prove the following rather technical lemmas. We use notation of Definition 8.

LEMMA 14. Let

$$F(x_1, \dots, x_n) := \bigvee_{1 \leq i_1 \leq m} a_{i_1}(x_1) \wedge b_{i_1}(x_1, \dots, x_n)$$

be a CAF and let $-\infty = r_0 < r_1 < \dots < r_l < r_{l+1} = \infty$ be such that all real roots of elements of $P_F \cap \mathbb{R}[x_1]$ are among $r_1, \dots, r_l$. Let $a(x_1)$ be either $x_1 = r_j$ for some $1 \leq j \leq l$, or $r_j < x_1 < r_{j+1}$ for some $0 \leq j \leq l$, and let C_a be the solution set of a . Then

$$\forall x_1 \in C_a \quad F(x_1, \dots, x_n) \Leftrightarrow G(x_1, \dots, x_n) \quad (3.1)$$

and one of the following two statements is true

1. There exist $1 \leq i_1 \leq m$ such that $C_a \subseteq C_{i_1}$ and $G(x_1, \dots, x_n) = b_{i_1}(x_1, \dots, x_n)$.
2. For all $1 \leq i_1 \leq m$, $C_a \cap C_{i_1} = \emptyset$ and $G(x_1, \dots, x_n) = \text{false}$

Moreover, given F and a , G can be found algorithmically.

PROOF. Let r be an algebraic number that appears in a_{i_1} . Then $r = \text{Root}_{x_1, p} f$ for some $f \in P_F$. Hence, $r = r_{j_0}$ for some $1 \leq j_0 \leq l$ and the value of j_0 can be determined algorithmically. If a is $x_1 = r_j$, then $C_a \cap C_{i_1} \neq \emptyset$ iff a_{i_1} is either $x_1 = r_j$ or $r_u < x_1 < r_v$ with $u < j < v$. In both cases $C_a \subseteq C_{i_1}$. If a is $r_j < x_1 < r_{j+1}$, then $C_a \cap C_{i_1} \neq \emptyset$ iff a_{i_1} is $r_u < x_1 < r_v$ with $u \leq j$ and $v \geq j+1$. In this case also $C_a \subseteq C_{i_1}$. Equivalence (3.1) follows from the statements (1) and (2). $\square$

LEMMA 15. Let $2 \leq k \leq n$, let

$$b_{i_1, \dots, i_{k-1}}(x_1, \dots, x_n) := \bigvee_{1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}} a_{i_1, \dots, i_k}(x_1, \dots, x_k) \wedge b_{i_1, \dots, i_k}(x_1, \dots, x_n)$$

be a level k cylindrical algebraic subformula of a CAF F , let $W = (W_1, \dots, W_n)$ be a projection sequence for P_F . Let $C \subseteq \mathbb{R}^{k-1}$ be a cell such that all polynomials of W_{k-1} have constant signs on C and $C \subseteq C_{i_1, \dots, i_{k-1}}$. Let $(c_1, \dots, c_{k-1}) \in C$ and let $d_1 < \dots < d_l$ be all real roots of $\{f(c_1, \dots, c_{k-1}, x_k) : f \in W_k\}$. For $1 \leq j \leq l$, let $r_j := \text{Root}_{x_k, p} f$, where $f \in W_k$ and d_j is the p -th root of $f(c_1, \dots, c_{k-1}, x_k)$. Let $a(x_1, \dots, x_k)$ be either $x_k = r_j$ for some $1 \leq j \leq l$, or $r_j < x_k < r_{j+1}$ for some $0 \leq j \leq l$, where $r_0 := -\infty$ and $r_{l+1} := \infty$ and let

$$C_a := \{(x_1, \dots, x_k) : (x_1, \dots, x_{k-1}) \in C \wedge a(x_1, \dots, x_k)\}$$

Then

$$\forall (x_1, \dots, x_k) \in C_a \quad b_{i_1, \dots, i_{k-1}}(x_1, \dots, x_n) \Leftrightarrow G(x_1, \dots, x_n) \quad (3.2)$$

and one of the following two statements is true

1. There exist $1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}$ such that

$$C_a \subseteq C_{i_1, \dots, i_{k-1}, i_k}$$

and

$$G(x_1, \dots, x_n) = b_{i_1, \dots, i_k}(x_1, \dots, x_n)$$

2. For all $1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}$

$$C_a \cap C_{i_1, \dots, i_{k-1}, i_k} = \emptyset$$

and

$$G(x_1, \dots, x_n) = \text{false}$$

Moreover, given $b_{i_1, \dots, i_{k-1}}$, a , $(c_1, \dots, c_{k-1})$, $d_1, \dots, d_l$ and the multiplicity of d_j as a root of f , for all $1 \leq j \leq l$ and $f \in W_k$, G can be found algorithmically.

PROOF. Let r be an algebraic function that appears in $a_{i_1, \dots, i_k}$. Then $r = \text{Root}_{x_k, p} f$ for some $f \in P_F$. By Definition 7, r is defined and continuous on C . Since W is a projection sequence for P_F , all factors of f that depend on x_k are elements of W_k . Hence, $r(c_1, \dots, c_{k-1}) = d_{j_0}$ for some $1 \leq j_0 \leq l$. Since d_{j_0} is the p -th of real roots of factors of f , multiplicities counted, if the multiplicity of d_j as a root of f is known for all $1 \leq j \leq l$ and $f \in W_k$, the value of j_0 can be determined algorithmically. Since all polynomials of W_{k-1} have constant signs on C , all elements of W_k that are not identically zero on C are delinable over C . Therefore, $r = r_{j_0}$ and $r_1 < \dots < r_l$ on C . If a is $x_k = r_j$, then $C_a \cap C_{i_1, \dots, i_{k-1}, i_k} \neq \emptyset$ iff $a_{i_1, \dots, i_k}$ is either $x_k = r_j$ or $r_u < x_k < r_v$ with $u < j < v$. In both cases $C_a \subseteq C_{i_1, \dots, i_{k-1}, i_k}$. If a is $r_j < x_k < r_{j+1}$, then $C_a \cap C_{i_1, \dots, i_{k-1}, i_k} \neq \emptyset$ iff $a_{i_1, \dots, i_k}$ is $r_u < x_k < r_v$ with $u \leq j$ and $v \geq j+1$. In this case also $C_a \subseteq C_{i_1, \dots, i_{k-1}, i_k}$. Equivalence (3.2) follows from the statements (1) and (2). $\square$

For simplicity we present a quantifier-free version of the algorithm. Extension to quantified formulas is straightforward and follows the ideas of [7].

ALGORITHM 16. (CAFCombine quantifier free)

Input: Cylindrical algebraic formulas

$$F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n)$$

and a Boolean formula $\Phi(p_1, \dots, p_m)$.

Output: A CAF $F(x_1, \dots, x_n)$ such that

$$F(x_1, \dots, x_n) \Leftrightarrow \Phi(F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n)) \quad (3.3)$$

1. Let $W := (W_1, \dots, W_n)$ be a projection sequence for $P_{F_1} \cup \dots \cup P_{F_m}$.
2. Let $r_1 < \dots < r_l$ be all real roots of elements of W_1 .
3. For $1 \leq i \leq l$, set $a_{2i}(x_1) := (x_1 = r_i)$ and $c_{1,2i} := r_i$.
4. For $0 \leq i \leq l$, set $a_{2i+1}(x_1) := (r_i < x_1 < r_{i+1})$ and pick $c_{1,2i+1} \in (r_i, r_{i+1}) \cap \mathbb{Q}$, where $r_0 := -\infty$ and $r_{l+1} := \infty$.
5. For $1 \leq i \leq 2l+1$

- (a) For $1 \leq j \leq m$, let G_j be the formula G found using Lemma 14 applied to F_j and a_i .
- (b) Let $\Psi := \Phi(G_1, \dots, G_m)$. If Ψ is true or false, set $b_i(x_1, \dots, x_n) := \Psi$.

(c) Otherwise set

$$b_i(x_1, \dots, x_n) := \text{Lift}((c_{1,i}), W, G_1, \dots, G_m, \Phi)$$

6. Return

$$F(x_1, \dots, x_n) := \bigvee_{1 \leq i \leq 2l+1} a_i(x_1, \dots, x_k) \wedge b_i(x_1, \dots, x_n)$$

Let us now describe the recursive subalgorithm *Lift* used in step 5(c). The subalgorithm requires its input to satisfy the following conditions.

1. $W = (W_1, \dots, W_n)$ is a projection sequence for $P_{F_1} \cup \dots \cup P_{F_m}$.
2. $(c_1, \dots, c_{k-1}) \in C$, $2 \leq k \leq n$ and $C \subseteq \mathbb{R}^{k-1}$ is a cell such that all polynomials of W_{k-1} have constant signs on C .
3. Each B_j is a level k cylindrical algebraic subformula of F_j or false.
4. C is contained in the intersection of support cells of all B_j that are not false.
5. $\Phi(p_1, \dots, p_m)$ is a Boolean formula.

ALGORITHM 17. (*Lift*)

Input: $(c_1, \dots, c_{k-1}) \in \mathbb{R}^{k-1}$, $W, B_1, \dots, B_m, \Phi$.

Output: A level k cylindrical algebraic subformula

$$b(x_1, \dots, x_n)$$

such that

$$\forall (x_1, \dots, x_{k-1}) \in C \quad b(x_1, \dots, x_n) \Leftrightarrow \Phi(B_1(x_1, \dots, x_n), \dots, B_m(x_1, \dots, x_n)) \quad (3.4)$$

1. Let $d_1 < \dots < d_l$ be all real roots of
$$\{f(c_1, \dots, c_{k-1}, x_k) : f \in W_k\}$$
2. For $1 \leq i \leq l$, let $r_i := \text{Root}_{x_k, p} f$, where $f \in W_k$ and d_i is the p -th root of $f(c_1, \dots, c_{k-1}, x_k)$.
3. For $f \in W_k$, if d_i is a root of $f(c_1, \dots, c_{k-1}, x_k)$, let $M(f, i)$ be its multiplicity, otherwise $M(f, i) := 0$.
4. For $1 \leq i \leq l$, set $a_{2i}(x_1, \dots, x_k) := (x_k = r_i)$ and $c_{k,2i} := d_i$.
5. For $0 \leq i \leq l$, set

$$a_{2i+1}(x_1, \dots, x_k) := (r_i < x_k < r_{i+1})$$

where $r_0 := -\infty$ and $r_{l+1} := \infty$, and pick $c_{k,2i+1} \in (d_i, d_{i+1}) \cap \mathbb{Q}$, where $d_0 := -\infty$ and $d_{l+1} := \infty$.

6. For $1 \leq i \leq 2l+1$

- (a) For $1 \leq j \leq m$, if $B_j = \text{false}$, set $G_j = \text{false}$, otherwise let G_j be the formula G found using Lemma 15 applied to $B_j, a_i, (c_1, \dots, c_{k-1}), d_1, \dots, d_l$ and M .
- (b) Let $\Psi := \Phi(G_1, \dots, G_m)$. If Ψ is true or false, set $b_i(x_1, \dots, x_n) := \Psi$.
- (c) Otherwise set $b_i(x_1, \dots, x_n)$ to

$$\text{Lift}((c_1, \dots, c_{k-1}, c_{k,i}), W, G_1, \dots, G_m, \Phi)$$

7. Return

$$b(x_1, \dots, x_n) := \bigvee_{1 \leq i \leq 2l+1} a_i(x_1, \dots, x_k) \wedge b_i(x_1, \dots, x_n)$$

PROOF. (Correctness of *CAFCombine*) Let us first show that inputs to *Lift* satisfy the required conditions. Condition (1) follows from step 1 of *CAFCombine*. If $k = 2$, the cell C is defined as a root or the open interval between two subsequent roots of polynomials of W_1 . For $k > 2$, the cell C is defined as a graph of a root or the set between graphs of two subsequent roots of polynomials of W_{k-1} over a cell on which W_{k-1} is delineable. This proves condition (2). Conditions (3) and (4) are guaranteed by Lemmas 14 and 15. Finally, (5) is satisfied, because Φ is always the same formula, given as input to *CAFCombine*.

To complete the proof we need to show the equivalences (3.3) and (3.4). Equivalence (3.4) follows from Lemma 15 and the fact that the sets

$$\{(x_1, \dots, x_k) : (x_1, \dots, x_{k-1}) \in C \wedge a_i(x_1, \dots, x_k)\}$$

are disjoint and their union is equal to $C \times \mathbb{R}$. Equivalence (3.3) follows from Lemma 14 and the fact that the sets $\{x_1 \in \mathbb{R} : a_i(x_1)\}$ are disjoint and their union is equal to $\mathbb{R}$. $\square$

4. IMPROVEMENT

The main idea behind the improvement presented in this section comes from the observation that if the projections on $\mathbb{R}$ of semialgebraic sets do not intersect then the intersection of the sets is empty and a CAF representing the union of the sets can be obtained by simple reordering of the disjunction of CAFs representing the sets. More generally, when lifting a cell in $C \subseteq \mathbb{R}$ in *CAFCombine* it suffices to work with a projection sequence for the set of defining polynomials of algebraic functions used in the description of a cell whose projection on $\mathbb{R}$ intersects C . As we will see in the last section, this leads to a significant performance improvement in practice.

ALGORITHM 18. (*CAFCombineI*)

Input: Cylindrical algebraic formulas

$$F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n)$$

and a Boolean formula $\Phi(p_1, \dots, p_m)$.

Output: A CAF $F(x_1, \dots, x_n)$ such that

$$F(x_1, \dots, x_n) \Leftrightarrow \Phi(F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n)) \quad (4.1)$$

1. Let $r_1 < \dots < r_l$ be all real roots of $(P_{F_1} \cup \dots \cup P_{F_m}) \cap \mathbb{R}[x_1]$.
2. For $1 \leq i \leq l$, set $a_{2i}(x_1) := (x_1 = r_i)$ and for $0 \leq i \leq l$, set $a_{2i+1}(x_1) := (r_i < x_1 < r_{i+1})$, where $r_0 := -\infty$ and $r_{l+1} := \infty$.
3. For $1 \leq i \leq 2l+1$

- (a) For $1 \leq j \leq m$, let G_j be the formula G found using Lemma 14 applied to F_j and a_i .
- (b) Let $j_1, \dots, j_s$ be all $1 \leq j \leq m$ for which G_j is neither true nor false.
- (c) Let $\Psi(p_{j_1}, \dots, p_{j_s})$ be the formula obtained from Φ by replacing p_j with G_j for all j for which G_j is true or false.

(d) If Ψ is true or false, set $H_i(x_1, \dots, x_n) := a_i \wedge \Psi$.

(e) Otherwise set $H_i(x_1, \dots, x_n)$ to

$$\text{CAFCombine}(a_i \wedge G_{j_1}, \dots, a_i \wedge G_{j_s}, \Psi)$$

4. Return $F(x_1, \dots, x_n) := \bigvee_{1 \leq i \leq 2l+1} H_i(x_1, \dots, x_n)$.

Correctness of the algorithm follows from Lemma 14, correctness of *CAFCombine* and the fact that the sets $\{x_1 \in \mathbb{R} : a_i(x_1)\}$ are disjoint and their union is equal to $\mathbb{R}$.

EXAMPLE 19. Let

$$\begin{aligned} f_1 &:= (x+1)^4 + y^4 - 4 \\ g_1 &:= (x+2)^2 + y^2 - 5 \\ f_2 &:= (x-1)^4 + y^4 - 4 \\ g_2 &:= (x-2)^2 + y^2 - 5 \end{aligned}$$

and let

$$\begin{aligned} A_1 &:= \{(x, y) \in \mathbb{R}^2 : f_1 < 0 \wedge g_1 < 0\} \\ A_2 &:= \{(x, y) \in \mathbb{R}^2 : f_2 < 0 \wedge g_2 < 0\} \end{aligned}$$

The following CAF's represent cell decompositions of A_1 and A_2 .

$$\begin{aligned} F_1(x, y) &:= r_1 < x < r_2 \wedge \text{Root}_{y,1} f_1 < y < \text{Root}_{y,2} f_1 \vee \\ &\quad x = r_2 \wedge \text{Root}_{y,1} f_1 < y < \text{Root}_{y,2} f_1 \vee \\ &\quad r_2 < x < r_4 \wedge \text{Root}_{y,1} g_1 < y < \text{Root}_{y,2} g_1 \\ F_2(x, y) &:= r_3 < x < r_5 \wedge \text{Root}_{y,1} g_2 < y < \text{Root}_{y,2} g_2 \vee \\ &\quad x = r_5 \wedge \text{Root}_{y,1} g_2 < y < \text{Root}_{y,2} g_2 \vee \\ &\quad r_5 < x < r_6 \wedge \text{Root}_{y,1} f_2 < y < \text{Root}_{y,2} f_2 \end{aligned}$$

where

$$\begin{aligned} r_1 &:= -1 - \sqrt{2} \approx -2.414 \\ r_2 &:= \text{Root}_{x,1} x^4 + 6x^3 + 10x^2 - 2x - 1 \approx -0.244 \\ r_3 &:= 2 - \sqrt{5} \approx -0.236 \\ r_4 &:= -2 + \sqrt{5} \approx 0.236 \\ r_5 &:= \text{Root}_{x,2} x^4 - 6x^3 + 10x^2 + 2x - 1 \approx 0.244 \\ r_6 &:= 1 + \sqrt{2} \approx 2.414 \end{aligned}$$

Compute a CAF representation of $A_1 \cap A_2$ (Figure 1) using *CAFCombineI*.

The input consists of F_1 , F_2 and $\Phi(p_1, p_2) := p_1 \wedge p_2$. The roots computed in step (1) are r_1 , r_2 , r_3 , r_4 , r_5 and r_6 . In step (3), for all $i \neq 7$, either G_1 or G_2 is false, and hence $\Psi = \text{false}$. For $i = 7$ the algorithm computes *CAFCombine*($a_7 \wedge G_1, a_7 \wedge G_2, \Phi$), where

$$\begin{aligned} a_7 \wedge G_1 &:= r_3 < x < r_4 \wedge \text{Root}_{y,1} g_1 < y < \text{Root}_{y,2} g_1 \\ a_7 \wedge G_2 &:= r_3 < x < r_4 \wedge \text{Root}_{y,1} g_2 < y < \text{Root}_{y,2} g_2 \end{aligned}$$

The projection sequence computed in step (1) is

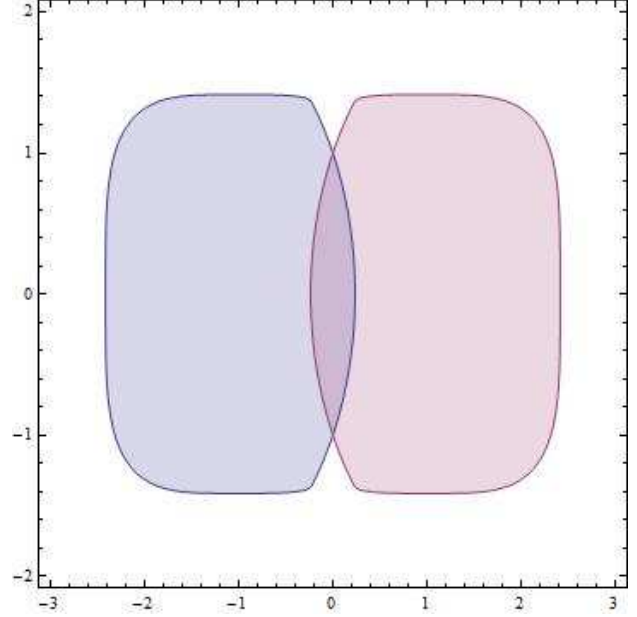
$$\begin{aligned} W_2 &:= \{g_1, g_2\} \\ W_1 &:= \{x, x^2 + 4x - 1, x^2 - 4x - 1\} \end{aligned}$$

The roots computed in step (2) are

$$-2 - \sqrt{5} < r_3 < 0 < r_4 < 2 + \sqrt{5}$$

In step (6), for all $i \notin \{5, 6, 7\}$ either G_1 or G_2 is false, and hence $\Psi = \text{false}$. The returned cell decomposition of

Figure 4.1: Sets A_1 and A_2



$A_1 \cap A_2$ consists of three cells constructed for $i \in \{5, 6, 7\}$.

$$\begin{aligned} F(x, y) &:= r_3 < x < 0 \wedge \text{Root}_{y,1} g_2 < y < \text{Root}_{y,2} g_2 \vee \\ &\quad x = 0 \wedge -1 < y < 1 \vee \\ &\quad 0 < x < r_4 \wedge \text{Root}_{y,1} g_1 < y < \text{Root}_{y,2} g_1 \end{aligned}$$

Note that the computation did not require including f_1 and f_2 in the projection set.

5. EMPIRICAL RESULTS

Algorithms *CAFCombine* and *CAFCombineI* have been implemented in C, as a part of the kernel of *Mathematica*. In the experiments we use two implementations CAD. The *Mathematica* 7 implementation returns cylindrical algebraic formulas. For methods that require polynomial output of CAD, we used *QEPCAD*, version B 1.53 [7, 1]. *QEPCAD* was called with the command line option $+N1000000000$. In the first two experiments each computation was given a time limit of 3600 seconds. In the third experiment each computation was given a time limit of 600 seconds. The experiments have been conducted on a 2.8 GHz Intel Xeon processor, with 72 GB of RAM available.

5.1 Solotareff problem (Example 3)

We run the first method in *Mathematica* and in *QEP-CAD*. Neither computation finished within the 3600 second time limit. With the second method, the computation of $P_4(r, a, b, u)$ using *QEPCAD* did not finish in 3600 seconds (*Mathematica* does not implement polynomial solution formula construction). The third method solves the problem in the total time of 7.57 seconds. The CAD computation which finds $F_1(r, a, b, u)$ takes 4.38 seconds and the CAD computation which finds $F_2(r, a, b, u)$ takes 0.20 seconds. The computation of

$$\exists u \in \mathbb{R} F_1(r, a, b, u) \wedge F_2(r, a, b, u)$$

with *CAFCombineI* computation takes 2.99 seconds. With *CAFCombine* without the improvement described in Section

4 the same computation takes 369 seconds. The computed solution is

$$r > 12 - 8\sqrt{2} \wedge a = \text{Root}_{a,2}f \wedge b = \text{Root}_{b,1}g$$

where

$$\begin{aligned} f = & 324a^4 + (324r^2 - 2016)a^3 + (108r^4 - 1128r^2 + \\ & 4576)a^2 + (12r^6 - 224r^4 + 1392r^2 - 4480)a - \\ & 15r^6 + 112r^4 - 608r^2 + 1600 \end{aligned}$$

and

$$\begin{aligned} g = & 27b^2 - ((18r - 36)a + 4r^3 - 6r^2 + 30r + 40)b - \\ & 4a^3 - (r^2 + 8r - 20)a^2 - (2r^3 - 18r^2 + 12r + 32)a + \\ & 3r^4 - 2r^3 + 3r^2 + 24r + 16 \end{aligned}$$

The solution is equivalent to the solution given in [6]. The equivalence can be proven using *CAFCombine* in 4.09 seconds.

5.2 Distance of roots of a cubic

Problem: Prove that the distance of two real roots of a monic cubic polynomial which maps $[-1, 1]$ into $[-1, 1]$ must be less than 3.

Method 1: Use the CAD algorithm to prove that the solution set of

$$\begin{aligned} \forall z \exists x \exists y \ x^3 + ax^2 + bx + c = 0 \wedge y^3 + ay^2 + by + c = 0 \wedge \\ x - y \geq 3 \wedge (-1 \leq z \leq 1 \Rightarrow -1 \leq z^3 + az^2 + bz + c \leq 1) \end{aligned}$$

is empty.

Method 2: Use the CAD algorithm to find a polynomial system of equations and inequalities $P_1(a, b, c)$ equivalent to

$$\exists x \exists y \ x^3 + ax^2 + bx + c = 0 \wedge y^3 + ay^2 + by + c = 0 \wedge x - y \geq 3$$

and a polynomial system of equations and inequalities $P_2(a, b, c)$ equivalent to

$$\forall z \ (-1 \leq z \leq 1 \Rightarrow -1 \leq z^3 + az^2 + bz + c \leq 1)$$

and then use the CAD algorithm to prove that the solution set of $P_1(a, b, c) \wedge P_2(a, b, c)$ is empty.

Method 3: Use the CAD algorithm to find a CAF $F_1(a, b, c)$ representing the solution set of

$$\exists x \exists y \ x^3 + ax^2 + bx + c = 0 \wedge y^3 + ay^2 + by + c = 0 \wedge x - y \geq 3$$

and a CAF $F_2(a, b, c)$ representing the solution set of

$$\forall z \ (-1 \leq z \leq 1 \Rightarrow -1 \leq z^3 + az^2 + bz + c \leq 1)$$

and then use *CAFCombineI* or *CAFCombine* to prove that the solution set of $F_1(a, b, c) \wedge F_2(a, b, c)$ is empty.

We run the first method in *Mathematica* and in *QEPCAD*. Neither computation finished within the 3600 second time limit.

For the second method, computation of $P_1(a, b, c)$ and $P_2(a, b, c)$ with *QEPCAD* took, respectively, 0.81 seconds and 533 seconds. Showing that the solution set of $P_1(a, b, c) \wedge P_2(a, b, c)$ is empty took 142 seconds with *Mathematica* and didn't finish in 3600 seconds with *QEPCAD*. Hence, the lowest total timing for the second method is 676 seconds.

With the third method, computation of $F_1(a, b, c)$ and $F_2(a, b, c)$ with *Mathematica* took, respectively, 0.12 seconds and 97.6 seconds. Showing that the solution set of $F_1(a, b, c) \wedge F_2(a, b, c)$ is empty took 3.03 seconds with *CAFCombineI*

Table 1: Unions and intersections of unit balls, $v = 2$

n	Union			Intersection			CAF
	CAD	CC	CCI	CAD	CC	CCI	
2	0.016	0.022	0.003	0.006	0.009	0.003	0.016
5	0.101	0.240	0.011	0.067	0.174	0.007	0.037
10	0.703	2.96	0.024	0.720	2.73	0.015	0.077
20	8.26	47.5	0.049	10.1	46.1	0.033	0.157
30	38.4	242	0.076	49.8	237	0.052	0.239

Table 2: Unions and intersections of unit balls, $v = 1$

n	Union			Intersection			CAF
	CAD	CC	CCI	CAD	CC	CCI	
2	0.113	0.136	0.100	0.032	0.034	0.031	0.027
5	0.599	0.899	0.394	0.146	0.167	0.008	0.036
10	2.48	5.51	0.886	1.08	2.65	0.016	0.075
20	15.6	57.7	1.87	12.4	44.4	0.033	0.154
30	57.0	268	2.86	57.7	233	0.050	0.235

and 29.3 seconds with *CAFCombine* without the improvement described in Section 4. Therefore, the lowest total timing form the third method is 101 seconds.

5.3 Unions and intersections of unit balls (Example 4)

Results of experiments computing $U_{v,n}$ and $I_{v,n}$ are given in Table 1 and Table 2. The timings are given in seconds. Columns marked *CAD* give the timings for computing $U_{v,n}$ and $I_{v,n}$ using the CAD algorithm (Method 1). The columns marked *CC* give the timings for *CAFCombine* computations without the improvement described in Section 4. The columns marked *CCI* give the timings for *CAFCombineI* computations. The column marked *CAF* gives the times for computing CAFs representing the unit balls. The total timings for the two versions of the second method are sums of entries in the *CAF* and *CC* or *CAF* and *CCI* columns.

5.4 Conclusions

The first two experiments show that for some quantifiers elimination problems partitioning the problem into subproblems, getting CAF descriptions for solution sets of the subproblems and then combining the results using *CAFCombine* is faster than either direct quantifier elimination using CAD or getting polynomial descriptions for solution sets of the subproblems and then computing the CAD of the combined results. The last experiment shows that for computation of a CAD for a set-theoretic combination of semialgebraic sets, computing a CAF representing each set and using *CAFCombineI* may be faster than direct CAD computation from polynomial description. The advantage of *CAFCombineI* is

Table 3: Unions and intersections of unit balls, $v = \frac{1}{2}$

n	Union			Intersection			CAF
	CAD	CC	CCI	CAD	CC	CCI	
2	0.327	1.12	0.312	0.196	0.598	0.183	0.029
5	10.0	32.4	8.24	1.86	0.220	0.052	0.048
10	42.2	144	23.3	6.50	2.84	0.034	0.096
20	175	> 600	53.0	40.5	45.8	0.070	0.189
30	426	> 600	82.7	144	237	0.108	0.282

Table 4: Unions and intersections of unit balls, $v = \frac{1}{4}$

n	Union			Intersection			CAF
	CAD	CC	CCI	CAD	CC	CCI	
2	0.324	1.37	0.305	0.215	0.874	0.191	0.032
5	39.3	293	34.6	14.3	55.4	5.80	0.052
10	534	> 600	395	84.1	3.35	0.042	0.090
20	> 600	> 600	> 600	258	47.4	0.089	0.249
30	> 600	> 600	> 600	> 600	241	0.134	0.380

greater if the number of intersections between the sets is smaller.

In all examples the version of *CAFCombine* algorithm with improvement described in Section 4 performed significantly better than the version without the improvement.

6. REFERENCES

- [1] C. W. Brown. An overview of qepcad b: a tool for real quantifier elimination and formula simplification. *J. JSSAC*, 10:13–22, 2003.
- [2] C. W. Brown. Qepcad b - a program for computing with semi-algebraic sets using cads. *ACM SIGSAM Bulletin*, 37:97–108, 2003.
- [3] B. Caviness and J. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*, New York, 1998. Springer Verlag.
- [4] C. Chen, M. M. Maza, B. Xia, and L. Yang. Computing cylindrical algebraic decomposition via triangular decomposition. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2009*, pages 95–102. ACM, 2009.
- [5] G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. *Lect. Notes Comput. Sci.*, 33:134–183, 1975.
- [6] G. E. Collins. Application of quantifier elimination to solotareff’s approximation problem. RISC Report Series 95-31, University of Linz, Austria, 1995.
- [7] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symbolic Comp.*, 12:299–328, 1991.
- [8] H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 1990*, pages 261–264. ACM, 1990.
- [9] D. Lazard. Solving kaltofen’s challenge on zolotarev’s approximation problem. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2006*, pages 196–203. ACM, 2006.
- [10] S. McCallum. An improved projection for cylindrical algebraic decomposition of three dimensional space. *J. Symbolic Comp.*, 5:141–161, 1988.
- [11] S. McCallum. An improved projection for cylindrical algebraic decomposition. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 242–268. Springer Verlag, 1998.
- [12] S. L. ojasiewicz. *Ensembles semi-analytiques*. I.H.E.S., 1964.
- [13] A. Strzeboński. Computing in the field of complex algebraic numbers. *J. Symbolic Comp.*, 24:647–656, 1997.
- [14] A. Strzeboński. Solving systems of strict polynomial inequalities. *J. Symbolic Comp.*, 29:471–480, 2000.
- [15] A. Strzeboński. Cylindrical algebraic decomposition using validated numerics. *J. Symbolic Comp.*, 41:1021–1038, 2006.
- [16] A. Tarski. *A decision method for elementary algebra and geometry*. University of California Press, 1951.

Black-Box/White-Box Simplification and Applications to Quantifier Elimination

Christopher W. Brown
Computer Science Department, Stop 9F
United States Naval Academy
572M Holloway Road
Annapolis, MD 21402, U.S.A.
wcbrown@usna.edu

Adam Strzeboński
Wolfram Research Inc.
100 Trade Centre Drive
Champaign, IL 61820, U.S.A.
adams@wolfram.com

ABSTRACT

This paper describes a new method for simplifying Tarski formulas. The method combines simplifications based purely on the factor structure of inequalities (“black-box” simplification) with simplifications that require reasoning about the factors themselves. The goal is to produce a simplification procedure that is very fast, so that it can be applied — perhaps many, many times — within other algorithms that compute with Tarski formulas without ever slowing them down significantly, but which also produces useful simplification in a substantial number of cases. The method has been implemented and integrated into implementations of two important algorithms: quantifier elimination by virtual term substitution, and quantifier elimination by cylindrical algebraic decomposition. The paper reports on how the simplification method has been integrated with these two algorithms, and reports experimental results that demonstrate how their performance is improved.

Categories and Subject Descriptors

G.4 [Mathematics of Computation]: Mathematical software

General Terms

Algorithms, Theory

Keywords

tarski formulas, simplification, quantifier elimination

1. INTRODUCTION

Computing with *semi-algebraic sets*, i.e. with sets defined by formulas consisting of boolean combinations of real polynomial equalities and inequalities, is a core subject in Computer Algebra. A variety of algorithms have been proposed and implemented for quantifier elimination, real system solv-

ing, satisfiability checking and other basic problems concerning semi-algebraic sets. Underlying our work is the hypothesis that these algorithms, in practice, would benefit from a simplification procedure that is always fast, and is often able to detect simplifications of formulas. In this paper we

1. propose a method for “fast simplification”,
2. describe how the method has been implemented by the second author and integrated with existing programs implementing two well-known algorithms for computing with semi-algebraic sets: cylindrical algebraic decomposition (CAD) [9, 1, 10] and quantifier elimination by virtual term substitution [16, 17], and
3. report experimental results that demonstrate how the performance of these program is affected by the integration of fast simplification.

All three of these represent new contributions.

1.1 Previous work

Simplification of boolean combinations of real polynomial equalities and inequalities, known as *Tarski formulas*, is a problem that has not received much attention [14, 12, 3, 18]. This is unfortunate, not only because simplification is an important problem in its own right, but also because *fast* simplification is needed to make decision or quantifier elimination methods efficient and not unduly sensitive to phrasings of input problems. In fact, this was a major motivation for [12]. In particular, their simplification methods were applied to intermediate results produced by the method of quantifier elimination by virtual term substitution in the Redlog system, which both speeds up the algorithm and reduces the size of its output. The other articles cited above apply a different philosophy: they use CAD to simplify a formula, which means that these approaches will almost certainly require a lot of time and a lot of memory, but can produce very simple formulas — often optimal in an appropriate sense. Our work is more in line with that of Dolzmann and Sturm.

In [4], the first author proposed simplifying a formula, in particular a conjunction, based solely on the factor structure¹ of the inequalities. Since each factor is treated as a

¹The term “factor structure” is used to refer to the boolean combination of monomial inequalities produced from a Tarski formula F by: 1) rewriting each atomic formula in F so that the left-hand side of the (in)equality is zero, 2) fully factoring each left-hand side, and 3) replacing each distinct factor with a distinct variable.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

“black box”, we refer to this as “black box simplification”. That paper gives efficient, polynomial time algorithms for discovering any sign conditions on individual factors that are implied, and for the satisfiability problem. Additionally, it is shown that finding an optimum simplification is NP-Hard. [5] describes an algorithm called *MinWtBasis* that provides optimum simplifications in this black box setting, provided all inequalities are non-strict. The algorithm is applicable in general, though its output only provides optimum simplification for the “non-strict part” of the input formula.

1.2 Fast simplification

This paper, like [12], is about what we call “fast simplification”. The specification for a fast simplification algorithm is simply that it takes a formula as input, produces an equivalent formula as output, and runs quickly — which for this work will mean polynomial time. While we would like the result to be simpler than the input, this specification does not address the issue. Thus, simply returning the input unchanged meets the specification. This is more than a bit unsatisfactory, and should be addressed.

The natural way to state the “the simplification problem” is to define a measure of complexity for formulas, and require a formula that minimizes the measure. However, this problem is utterly intractable for the obvious measure of formula length (and there is no reason to believe that the problem would be easier for any other meaningful measure). Even for extremely simple cases the problem is NP-Complete or complete in even higher complexity classes [4, 7]. Simplification based on CAD [14, 3, 18] is able to produce very short formulas, but 1) CAD computation is inherently doubly exponential in the number of variables [11, 6], and 2) even these methods do not produce a result that minimizes any meaningful metric on formulas.

Another approach would be to ask for “minimal” formulas, meaning that removing one or more inequalities always results in something that is not equivalent to the original formula. This problem is, clearly, at least as hard as satisfiability for Tarski formulas, and the only algorithms we know for solving the satisfiability problem run in time that is at least exponential in the number of variables, and high-degree polynomial in other parameters when the number of variables is fixed.

Thus, a specification for simplification that requires a result that minimizes some metric, or is minimal in the sense of containing no redundant or unnecessary inequalities, leads to algorithms with high time complexities in theory, and prohibitive running times in practice. So we are forced to leave the specification in its admittedly unsettling state, and evaluate fast simplification algorithms by how effective they are at helping other algorithms compute more quickly and produce smaller solutions.

1.3 Organization of this paper

The remainder of this paper is organized as follows: Section 2 defines the basic notation used throughout the paper, including our somewhat unusual representation of formulas. Section 3 describes the black-box simplification component of our simplification procedure. Section 4 describes what we term “white-box” algorithms, which actually compute algebraically with factors in a formula, rather than simply treating them as atomic, indivisible objects. In Section 5 we describe our primary algorithm, *Simplify*, which com-

bines black-box and white-box simplifications symbiotically to simplify input. Section 6 describes how *Simplify* has been integrated into two well-known and important algorithms for computing with semi-algebraic sets: CAD construction and quantifier elimination by virtual term substitution. Experimental results showing how these algorithms have been improved by making use of *Simplify* are presented in Section 7. Finally, in Section 8 summarize this paper’s contribution and look ahead to future work.

2. NOTATION AND REPRESENTATION OF FORMULAS

In the remainder of this paper, we present algorithms for manipulating formulas. Here we define our notation and the representation we use for formulas. We write each inequality with left-hand side zero. Thus, we might as well replace the usual binary relational operators with unary operators.

DEFINITION 1. Let OP be the set of the following eight unary operators mapping $\mathbb{R}$ into $\{false, true\}$.

1. $NOOP(x) := false$
2. $LTOP(x) := true \text{ if } x < 0, \text{ else } false$
3. $LEOP(x) := true \text{ if } x \leq 0, \text{ else } false$
4. $GTOP(x) := true \text{ if } x > 0, \text{ else } false$
5. $GEOP(x) := true \text{ if } x \geq 0, \text{ else } false$
6. $EQOP(x) := true \text{ if } x = 0, \text{ else } false$
7. $NEOP(x) := true \text{ if } x \neq 0, \text{ else } false$
8. $ALOP(x) := true$

We view a set $Q \subset \mathbb{Z}[x_1, \dots, x_n]$ and a function $\alpha : Q \rightarrow OP$ as defining the semi-algebraic set of all points x satisfying

$$\bigwedge_{q \in Q} \alpha(q)(q(x)).$$

Converting this representation into the equivalent conjunctions-of-inequalities and back is, of course, trivial. For example,

$$Q = \{x + y, x - y\}, \quad \alpha(q) = \begin{cases} LTOP & q = x + y \\ GEOP & q = x - y \end{cases}$$

represents $x + y < 0 \wedge x - y \geq 0$.

We define the binary operations $+, \cdot, \wedge : OP \times OP \rightarrow OP$ as:

$$\begin{aligned} (\alpha + \beta)(z) &:= true \text{ if } \exists x, y [z = x + y \wedge \alpha(x) \wedge \beta(y)], \text{ else } false \\ (\alpha \beta)(z) &:= true \text{ if } \exists x, y [z = xy \wedge \alpha(x) \wedge \beta(y)], \text{ else } false \\ (\alpha \wedge \beta)(z) &:= true \text{ if } \alpha(z) \wedge \beta(z), \text{ else } false \end{aligned}$$

and we define the unary operator $SQ : OP \rightarrow OP$ as:

$$SQ(\alpha)(z) := true \text{ if } \exists x [z = x^2 \wedge \alpha(x)], \text{ else } false$$

Although it is not obvious that, as defined, these operations only produce predicates from the set OP defined above, verifying the fact is not difficult. The following two definitions will also be needed in the sections that follow.

We define $sgn : \mathbb{R} \rightarrow OP$ by

$$sgn(x) := \begin{cases} LTOP & x < 0 \\ EQOP & x = 0 \\ GTOP & x > 0 \end{cases}$$

DEFINITION 2. Let $PP(x_1, \dots, x_n)$ denote the set of non-constant primitive polynomials in $\mathbb{Z}[x_1, \dots, x_n]$ with positive coefficients at the leading monomials with respect to the lexicographic order.

Define the *stronger than* relation on OP as follows: $a \in OP$ is stronger than $b \in OP$ provided that $a \neq b$ and $\forall x[a(x) \Rightarrow b(x)]$. Note that, based on this definition, for any subset $S \subseteq \mathbb{R}$ there is an unique *strongest* element of OP satisfied by every element of S .

3. BLACK BOX SIMPLIFICATION

Black box simplification is simplification based on the factor structure of a formula, without reasoning about the actual polynomials that comprise the factors. Algorithms and results related to this kind of simplification are described in [4, 5]. The remainder of this section assumes familiarity with those papers, however we highlight the following results:

- the black-box satisfiability problem can be solved in low-degree polynomial time,
- optimum black-box simplification is NP-Hard,
- all sign conditions of factors implied by the factor structure of the original formula can be discovered in low-degree polynomial time, and
- optimum black-box simplification for formulas containing only non-strict inequalities can be done in polynomial time

The above references do not propose a single, complete black-box simplification algorithm, so we provide a brief description of how we put the pieces from those papers together to produce a concrete algorithm.

Let $f_1, \dots, f_k$ be the distinct factors appearing in a conjunction F . Assume that $f_1, \dots, f_r$ appear in strict inequalities, while $f_{r+1}, \dots, f_k$ do not. We refer to these as the “strict” and “non-strict” factors, respectively. The algorithm MinWtBasis described in [5], produces an equivalent formula F' in which the sum over each inequality of the number of non-strict factors appearing in the inequality is minimized. More formally, writing F' as

$$\bigwedge_{i=1}^m \prod_{j=1}^r x_j^{e_{i,j}} \sigma_i 0,$$

MinWtBasis ensures that F' is a formula equivalent to F for which the basis weight function

$$wt(F') = \sum_{i=1}^m \sum_{j=r+1}^k \text{sgn}(e_{i,j})$$

is minimal. The algorithm runs in polynomial time.

One would like to produce from F' an equivalent formula in which the sum of the number of non-strict factors appearing in each inequality is minimized, but this is precisely what is proven in [4] to be intractable. However, what can be done in polynomial time is to guarantee that if the sign of a strict factor is implied by the factor structure, then the sign condition on that factor appears explicitly in the output formula. That is what our algorithm BlackBox does.

The correctness of Algorithm BlackBox follows easily from [4, 5], as does the fact that the running time is polynomial in the sizes of P and Q .

Algorithm 1 BlackBox

Input: Finite sets $P, Q \subseteq PP(x_1, \dots, x_n)$ such that each element of Q is a product of elements of P , $\alpha : P \ni p \rightarrow \alpha_p \in OP$ and $\beta : Q \ni q \rightarrow \beta_q \in OP$

Output: $\bar{Q} \subseteq PP(x_1, \dots, x_n)$, $\bar{\alpha} : P \ni p \rightarrow \bar{\alpha}_p \in OP$, $\bar{\beta} : \bar{Q} \ni q \rightarrow \bar{\beta}_q \in OP$ and $unsat \in \{\text{false}, \text{true}\}$ such that if $unsat$ is true, the formula F defined by

$$F := \bigwedge_{p \in P} \alpha_p(p) \wedge \bigwedge_{q \in Q} \beta_q(q)$$

is unsatisfiable, otherwise each element of $\bar{Q}$ is a product of elements of P and

$$F \Leftrightarrow \bigwedge_{p \in P} \bar{\alpha}_p(p) \wedge \bigwedge_{q \in \bar{Q}} \bar{\beta}_q(q)$$

furthermore,

- the sum over the elements of $\bar{Q}$ of the number of non-strict factors is minimum,
- if, for $p \in P$, $LTOP(p)$ (resp. $GTOP(p)$) is implied (in the blackbox sense) by F , then $\bar{\alpha}_p = LTOP$ (resp. $GTOP$),
- if, for $p \in P$, $EQOP(p)$ is implied (in the blackbox sense) by F then $\bar{\alpha}_p = EQOP$, and
- if neither of the other cases apply to p and $LEOP(p)$ (resp. $GEOP(p)$) is implied (in the blackbox sense) by F , $\bar{\alpha}_p = LEOP$ (resp. $GEOP$)

- 1: set $unsat := \text{false}$
 - 2: apply MinWtBasis to compute F' as described above
 - 3: let $u_1 \oplus v_1, \dots, u_j \oplus v_j$ be the vector representation of F' (see [4, 5]), w.l.o.g. assume $v_i = 0$ exactly for $1 \leq i \leq s$.
 - 4: let M be the matrix over $GF(2)$ with rows $u_1, \dots, u_s$.
 - 5: let M' be the reduced row-echelon form of M , w.l.o.g. assume the first t rows of M' are the rows that contain exactly one non-zero entry (excluding the last column).
 - 6: if $[0, \dots, 0, 1]$ is a row vector of M' , set $unsat := \text{true}$ and return
 - 7: let M'' be the matrix produced by row reducing each row of M by the first t rows of M' , removing any zero rows, and prepending the first t rows of M'
 - 8: **for all** i in $\{s+1, \dots, j\}$ **do**
 - 9: let u'_i be u_i reduced by the first t rows of M'
 - 10: let u''_i be u_i reduced by all the rows of M'
 - 11: let u^*_i be whichever of u'_i and u''_i has fewer non-zero entries (discounting the last entry)
 - 12: **end for**
 - 13: let M^* be whichever of M' and M'' has fewer non-zero entries (excluding the last column)
 - 14: let $W = \{u^*_i \oplus v_i \mid s < i \leq j\} \cup \{u \oplus 0 \mid u \text{ is a row of } M^*\}$
 - 15: let F'' be the formula equivalent of W
 - 16: let $\bar{Q}$, $\bar{\alpha}$ and $\bar{\beta}$ be the representation of formula F''
-

4. WHITE BOX ALGORITHMS

In “white-box simplification”, we seek to deduce new information on the signs of factors based on our current information on the signs of variables and of other factors. Here are a few examples of such deductions:

1. given $1 + y^2$, deduce $1 + y^2 > 0$
2. given $x^2 + y^2$ and $y \neq 0$ deduce $x^2 + y^2 > 0$
3. given $1 - x + y^2$ and $2x - 1 < 0$ deduce $1 - x + y^2 > 0$

These kinds of deductions can be used to simplify formulas. In this section we describe an algorithm called WhiteBox that makes deductions like these. Its use in simplification is discussed on a later section.

The more time one is willing to spend on deducing signs of factors, the more complete the information obtained. However, given the goal of fast simplification, we restrict ourselves to simple kinds of deductions that can be discovered (or found not to exist) quickly. In fact there are essentially only two kinds of deductions we consider. The first is the following: if, based on information on the signs of variables, one can deduce that every monomial in a polynomial p is non-negative (resp. non-positive), $p \geq 0$ (resp. $p \leq 0$). If, additionally, some monomial is seen to be positive (resp. negative) the deduced inequalities can be made strict. Finally, if all monomials can be deduced to be zero, $p = 0$ can be deduced. The algorithm PolynomialSign realizes this scheme, relying on algorithm MonomialSign to deduce signs of monomials based on information about the signs of variables.

Algorithm 2 MonomialSign

Input: Power product $M = x_1^{e_1} \dots x_n^{e_n}$ and $\alpha_1, \dots, \alpha_n \in OP$.
Output: β , the strongest element of OP such that $\alpha_1(x_1) \wedge \dots \wedge \alpha_n(x_n) \Rightarrow \beta(M)$
1: set $\beta := GTOP$.
2: **for all** $1 \leq i \leq n$ **do**
3: if e_i is even set $\beta := SQ(\alpha_i)\beta$, else set $\beta := \alpha_i\beta$
4: **end for**
5: return β

Algorithm 3 PolynomialSign

Input: Polynomial $p = a_1M_1 + \dots + a_kM_k$, where the M_i are power products over $x_1, \dots, x_n$ and $\alpha_1, \dots, \alpha_n \in OP$
Output: $\beta \in OP$ such that $\alpha_1(x_1) \wedge \dots \wedge \alpha_n(x_n) \Rightarrow \beta(p)$
1: set $\beta := EQOP$
2: **for all** $1 \leq i \leq k$ **do**
3: set $\beta := sgn(a_i)\text{MonomialSign}(M_i; \alpha_1, \dots, \alpha_n) + \beta$
4: **end for**
5: return β

It should be clear that algorithms MonomialSign and PolynomialSign are correct, and that their running times are polynomial, but it is worth recognizing that PolynomialSign could always return $ALOP$ and in so doing meet its specification. One might be tempted to tighten the specification to require that β is the strongest element of OP that meets the existing specification. However, this would leave us with a computationally difficult problem, which conflicts with the goal of fast simplification. The effectiveness of PolynomialSign in producing stronger results than the vacuous $ALOP$ is demonstrated in the experimental section of this paper by the practical value of the algorithms that rely upon it. Finally, we point out that PolynomialSign does produce the strongest possible result when no two monomials in p share a common variable.

The second kind of deduction is a bit less obvious. Suppose p and q are factors, and $q > 0$ is known. Suppose further that for some positive constant t we deduce (given some information on the signs of variables) that $p + tq < 0$. We may

then deduce $p < 0$. This should be clear. The algorithm DeduceSign generalizes this to all possible combinations of sign conditions on q , t , and deduced signs of $p + tq$. It relies on the algorithm FindIntervals, which (roughly) provides intervals I_1 and I_2 such that for all $t \in I_1$ we can deduce $p + tq \geq 0$, and for all $t \in I_2$ we can deduce $p + tq \leq 0$.

Algorithm 4 FindIntervals

Input: Polynomials $p = a_1M_1 + \dots + a_kM_k$ and $q = b_1M_1 + \dots + b_kM_k$ and $\alpha_1, \dots, \alpha_n \in OP$
Output: Intervals I_1 and I_2 and *strict* $\in \{false, true\}$ such that, letting F denote $\alpha_1(x_1) \wedge \dots \wedge \alpha_n(x_n)$,

$$F \Rightarrow \forall t \in I_1[p + tq \geq 0] \wedge \forall t \in I_2[p + tq \leq 0]$$

and if *strict*

$$F \Rightarrow \forall t \in \text{int}(I_1)[p + tq > 0] \wedge \forall t \in \text{int}(I_2)[p + tq < 0]$$

where *int* denotes the topological interior.

1: Set $I_1 := \mathbb{R}$, $I_2 := \mathbb{R}$ and *strict* = *false*.
2: **for all** $1 \leq i \leq k$ **do**
3: set $s := \text{MonomialSign}(M_i; \alpha_1, \dots, \alpha_n)$
4: **if** $s = NOOP$ **then** set $I_1 := \emptyset$ and $I_2 := \emptyset$
5: **if** $s = LTOP$ **then** set *strict* := *true*
6: **if** $s = LEOP$ or $LTOP$ **then**
7: **if** $b_i = 0 \wedge a_i < 0$ set $I_2 := \emptyset$
8: **if** $b_i = 0 \wedge a_i > 0$ set $I_1 := \emptyset$
9: **if** $b_i < 0$
 set $I_1 := I_1 \cap [-\frac{a_i}{b_i}, \infty)$ and $I_2 := I_2 \cap (-\infty, -\frac{a_i}{b_i}]$
10: **if** $b_i > 0$
 set $I_1 := I_1 \cap (-\infty, -\frac{a_i}{b_i}]$ and $I_2 := I_2 \cap [-\frac{a_i}{b_i}, \infty)$
11: **end if**
12: **if** $s = GTOP$ **then** set *strict* := *true*
13: **if** $s = GEOP$ or $GTOP$ **then**
14: **if** $b_i = 0 \wedge a_i < 0$ set $I_1 := \emptyset$
15: **if** $b_i = 0 \wedge a_i > 0$ set $I_2 := \emptyset$
16: **if** $b_i < 0$
 set $I_1 := I_1 \cap (-\infty, -\frac{a_i}{b_i}]$ and $I_2 := I_2 \cap [-\frac{a_i}{b_i}, \infty)$
17: **if** $b_i > 0$
 set $I_1 := I_1 \cap [-\frac{a_i}{b_i}, \infty)$ and $I_2 := I_2 \cap (-\infty, -\frac{a_i}{b_i}]$
18: **end if**
19: **if** $s = NEOP$ or $ALOP$ **then**
20: **if** $b_i = 0 \wedge a_i \neq 0$ set $I_1 := \emptyset$ and $I_2 := \emptyset$
21: else set $I_1 := I_1 \cap \{-\frac{a_i}{b_i}\}$ and $I_2 := I_2 \cap \{-\frac{a_i}{b_i}\}$
22: **end if**
23: **end for**
24: return I_1 , I_2 and *strict*.

FindIntervals operates by iterating over the power products of p and q , and refining the intervals I_1 and I_2 at each step. For instance, if the coefficient of power product M_i for p is 5, and for q is 8, and if the strongest deduction that can be made about the sign of M_i is $NOOP$, then the only way we can be sure that $p + xq \geq 0$ (resp. ≤ 0) is if $x = -5/8$, so that the M_i terms in $p + xq$ cancel. This is precisely what happens in line 21 of FindIntervals. Verifying the correctness of FindIntervals just requires checking each of these cases, which are each, individually, quite simple. The fact that it runs in time polynomial in the size of its input is self-evident. It can be shown that, in fact, I_1 exactly those value t for which $\text{PolynomialSign}(p + tq; \alpha_1, \dots, \alpha_n)$ returns $GTOP$ or $GEOP$. The analogous results hold for I_2 .

The algorithm DeduceSign combines the information re-

turned by FindIntervals about the sign of $p + xq$ for various values of q with what is known about the sign of q to determine what may be deduced about the sign of p . Once again, we provide no proof of correctness, since such a proof requires verifying a large number of very simple cases. In fact, the first author, implemented this algorithm as a table lookup in a table whose entries were generated by automatic quantifier elimination. The algorithms running time is clearly polynomial.

Algorithm 5 DeduceSign

Input: Polynomials $p = a_1M_1 + \dots + a_kM_k$ and $q = b_1M_1 + \dots + b_kM_k$ and $\alpha_1, \dots, \alpha_n, \beta \in OP$
Output: $\gamma \in OP$ s.t. $\alpha_1(x_1) \wedge \dots \wedge \alpha_n(x_n) \wedge \beta(q) \Rightarrow \gamma(p)$
1: set $\gamma := ALOP$
2: set $(I_1, I_2, strict) := FindIntervals(p; q; \alpha_1, \dots, \alpha_n)$
3: **if** $\beta = LTOP$ **then**
4: If $I_1 \cap \mathbb{R}_+ \neq \emptyset$, set $\gamma := GTOP$, else if $0 \in I_1$ set $\gamma := GEOP$.
 If $I_2 \cap \mathbb{R}_- \neq \emptyset$, set $\gamma := \gamma \wedge LTOP$, else if $0 \in I_2$ set $\gamma := \gamma \wedge LEOP$.
5: **else if** $\beta = LEOP$ **then**
6: If $strict$ and $int(I_1) \cap \mathbb{R}_+ \neq \emptyset$, set $\gamma := GTOP$, else if $I_1 \cap (\mathbb{R}_+ \cup \{0\}) \neq \emptyset$ set $\gamma := GEOP$.
 If $strict$ and $int(I_2) \cap \mathbb{R}_- \neq \emptyset$, set $\gamma := \gamma \wedge LTOP$, else if $I_2 \cap (\mathbb{R}_- \cup \{0\}) \neq \emptyset$ set $\gamma := \gamma \wedge LEOP$.
7: **else if** $\beta = GTOP$ **then**
8: If $I_1 \cap \mathbb{R}_- \neq \emptyset$, set $\gamma := GTOP$, else if $0 \in I_1$ set $\gamma := GEOP$.
 If $I_2 \cap \mathbb{R}_+ \neq \emptyset$, set $\gamma := \gamma \wedge LTOP$, else if $0 \in I_2$ set $\gamma := \gamma \wedge LEOP$.
9: **else if** $\beta = GEOP$ **then**
10: If $strict$ and $int(I_1) \cap \mathbb{R}_- \neq \emptyset$, set $\gamma := GTOP$, else if $I_1 \cap (\mathbb{R}_- \cup \{0\}) \neq \emptyset$ set $\gamma := GEOP$.
 If $strict$ and $int(I_2) \cap \mathbb{R}_+ \neq \emptyset$, set $\gamma := \gamma \wedge LTOP$, else if $I_2 \cap (\mathbb{R}_+ \cup \{0\}) \neq \emptyset$ set $\gamma := \gamma \wedge LEOP$.
11: **else if** $\beta = EQOP$ **then**
12: If $strict$ and $int(I_1) \neq \emptyset$, set $\gamma := GTOP$, else set $\gamma := GEOP$.
 If $strict$ and $int(I_2) \neq \emptyset$, set $\gamma := \gamma \wedge LTOP$, else set $\gamma := \gamma \wedge LEOP$.
13: **end if**
14: return γ .

As an example of how DeduceSign works, consider applying it to $p = 1 - x + y^2$, $q := 2x - 1$, $\alpha_x = ALOP$, $\alpha_y = ALOP$, $\beta = LTOP$, (i.e. example three from the beginning of this section). FindIntervals applied to p, q, α_x, α_y yields $I_1 = [1/2]$, $I_2 = \emptyset$, $strict = true$, which means that $p + 1/2q > 0$. Since $\beta = LTOP$ and $I_1 \cap \mathbb{R}_+ \neq \emptyset$, line 4 of DeduceSign sets $\gamma = GTOP$, i.e. we have deduced that $p > 0$.

Finally, in what follows, it will be convenient to have a procedure that discovers all such WhiteBox deductions for a formula.

The algorithm DeduceAll takes a factor p and what is known about the signs of other factors and variables and returns the strongest condition on the sign of p that it is able to deduce. The algorithm WhiteBox returns the strongest conditions it can deduce on the signs of all factors and variables appearing in its input.

The correctness of DeduceAll and WhiteBox is easily established. The algorithms essentially call DeduceAll to do

Algorithm 6 DeduceAll

Input: $p \in \mathbb{R}[x_1, \dots, x_n]$, $Q \subseteq \mathbb{R}[x_1, \dots, x_n]$, $\alpha_1, \dots, \alpha_n \in OP$ and $\beta : Q \ni q \rightarrow \beta_q \in OP$.
Output: $\gamma \in OP$ such that

$$\alpha_1(x_1) \wedge \dots \wedge \alpha_n(x_n) \wedge \forall q \in Q \beta_q(q) \Rightarrow \gamma(p)$$

1: set $\gamma := ALOP$
2: **for all** $q \in Q$ such that β_q is not *NEOP* or *ALOP* **do**
3: set $\gamma := \gamma \wedge DeduceSign(p; q; \alpha_1, \dots, \alpha_n; \beta_q)$.
4: if $\gamma = NOOP$, return *NOOP*.
5: **end for**
6: return γ .

Algorithm 7 WhiteBox

Input: $P, Q \subseteq \mathbb{R}[x_1, \dots, x_n]$, $\alpha_1, \dots, \alpha_n \in OP$, $\beta : P \ni p \rightarrow \beta_p \in OP$ and $\gamma : Q \ni q \rightarrow \gamma_q \in OP$
Output: $\bar{\alpha}_1, \dots, \bar{\alpha}_n \in OP$, $\bar{\beta} : P \ni p \rightarrow \bar{\beta}_p \in OP$ and $unsat \in \{false, true\}$ such that, defining F as

$$F := \alpha_1(x_1) \wedge \dots \wedge \alpha_n(x_n) \wedge \forall p \in P \beta_p(p) \wedge \forall q \in Q \gamma_q(q),$$

if *unsat* is true, F is unsatisfiable, otherwise

$$F \Leftrightarrow \bar{\alpha}_1(x_1) \wedge \dots \wedge \bar{\alpha}_n(x_n) \wedge \forall p \in P \bar{\beta}_p(p) \wedge \forall q \in Q \gamma_q(q)$$

1: set *unsat* := *false*. For $1 \leq i \leq n$, set $\bar{\alpha}_i := \alpha_i$. For each $p \in P$, set $\bar{\beta}_p := \beta_p$.
2: Set $R := P \cup Q$. For each $r \in R$, if $r \in P$, set $\delta_r := \beta_r$, else set $\delta_r := \gamma_r$.
3: **for all** $1 \leq i \leq n$ **do**
4: set $\bar{\alpha}_i := \bar{\alpha}_i \wedge DeduceAll(x_i; R; \bar{\alpha}_1, \dots, \bar{\alpha}_n; \delta)$.
5: if $\bar{\alpha}_i = NOOP$, set *unsat* := *true*
6: **end for**
7: **for all** $p \in P$ **do**
8: set $\bar{R} := R \setminus \{p\}$ and $\bar{\delta} := \delta \upharpoonright \bar{R}$.
9: set $\bar{\beta}_p := \bar{\beta}_p \wedge DeduceAll(p; \bar{R}; \bar{\alpha}_1, \dots, \bar{\alpha}_n; \bar{\delta})$.
10: if $\bar{\beta}_p = NOOP$, set *unsat* := *true*
11: **end for**
12: return $\bar{\alpha}_1, \dots, \bar{\alpha}_n, \bar{\beta}, unsat$

their work. That they run in time polynomial in the input size is also clear, since their loops simply iterate over their inputs.

5. BLACK-BOX/WHITE-BOX SIMPLIFICATION

Black-box simplification and white-box simplification inform one another. Black-box simplification discovers signs of factors if they are implied by the factor structure of the input. White-box simplification relies on having some sign information on factors in order to make deductions. In turn, stronger sign-conditions on factors then allows more Black-box simplification. Our approach is, more or less, to run the two algorithms in succession, until neither one makes new deductions. The complete algorithm is called “Simplify”. A broad outline grouping detailed steps of the full algorithm to a few conceptual steps is given below:

1. initialization, including extracting explicit sign conditions on variables (lines 1–9)
2. black-box simplification (lines 12–19)
3. white-box simplification (lines 20–25)

4. if new information has been deduced in Steps 2 or 3, goto Step 2 (line 26)
5. remove sign conditions on factors that are implied by other information (lines 28–34)
6. collect sign information into a new formula (lines 35,36)

Space does not permit a detailed proof of correctness or complexity analysis for algorithm Simplify, but a proof of termination (and in fact of polynomial running time) and some discussion of how the algorithm works is warranted. Although care is required to formulate the Simplify algorithm and implement it correctly, its correctness (assuming the correctness of the sub-algorithms PolynomialSign, DeduceAll, BlackBox and WhiteBox) relatively straightforward to establish. As the outline given above indicates, the algorithm is essentially a loop that applies BlackBox and WhiteBox in turn, repeatedly, to deduce stronger and stronger sign conditions on the variables and factors that appear in the input (lines 10–26). The process ends when no more strengthenings are deduced — or when sign condition *NOOP* is deduced, which implies that the original input is unsatisfiable. The remainder of the algorithm (lines 27–37) combines the strongest known sign conditions to create a new formula, which is represented by Q, τ .

The Simplify terminates is not completely obvious, since it is not clear when the algorithm breaks out of the repeat/until loop. The variable *changed* is set to *false* at the beginning of each iteration, and another iteration is initiated only if *changed* is subsequently set to *true*. This may only occur at two places in the algorithm: lines 16 and 24. Variable *changed* is set to *true* at line 16 only when $\bar{\delta} \neq \delta$, and at line 24 only when $(\bar{\alpha}_1, \dots, \bar{\alpha}_n) \neq (\alpha_1, \dots, \alpha_n)$ or $\bar{\beta} \neq \beta$. In the loop, α_i is the sign condition that, prior to the call of WhiteBox on line 20, variable x_i is known to satisfy; $\bar{\alpha}_i$ is the sign condition known after the call to WhiteBox. Similarly, β_p is the sign condition that, prior to the call of WhiteBox on line 20, the (non-variable) factor p is known to satisfy; $\bar{\beta}_p$ is the sign condition known after the call to WhiteBox. Thus, *changed* is set to *true* on line 24 if and only if the call to WhiteBox strengthens the known sign condition on some factor p or variable x_i . In the same way, δ_p is the sign condition that, prior to the call to BlackBox, factor p is known to satisfy, and $\bar{\delta}_p$ is the sign condition known after the call to BlackBox. Thus, *changed* is set to *true* on line 16 if and only if the call to BlackBox strengthens the known sign condition on some factor p , or $\bar{P}_3 \neq P_3 \vee \bar{\gamma} \neq \gamma$, which means that BlackBox returned a different set of multi-factor (in)equalities. If MinWtBasis is implemented so that if it cannot strictly improve the weight $wt(F)$ of the input formula F , it returns F unaltered (which is trivial to ensure), then BlackBox is idempotent, meaning that applying it twice yields the same result as applying it once. Thus, it is impossible to have two consecutive iterations in which *changed* is set to *true* without some irreducible factor or variable having its sign condition strengthened. So at least one in every two iterations (not including the last one) strengthens the sign condition of some irreducible factor or variable. Since the number of irreducible factors (i.e. $|P_1|$) and the number of variables (i.e. n) is fixed for the duration of the algorithm, and since the longest chain of sign strengthenings is four (e.g. *ALOP*, *GEOP*, *GTOP*, *NOOP*) the number of iterations is at most $1 + 8(n + |P_1|)$. Thus, the algorithm is

Algorithm 8 Simplify

Input: A finite set $P \subseteq PP(x_1, \dots, x_n)$ and $\sigma : P \ni p \rightarrow \sigma_p \in OP \setminus \{NOOP\}$

Output: A finite set $Q \subseteq PP(x_1, \dots, x_n)$, $\tau : Q \ni q \rightarrow \tau_q \in OP$ and *unsat* $\in \{false, true\}$ such that (defining F as $\forall p \in P \sigma_p(p)$) if *unsat* = *true*, F is unsatisfiable, otherwise $F \Leftrightarrow \forall q \in Q \tau_q(q)$

- 1: Set $Q := P$, $\tau := \sigma$ and *unsat* := *false*.
- 2: For $1 \leq i \leq n$ set $\alpha_i := \sigma_{x_i}$ if $x_i \in P$ else *ALOP*.
- 3: Let $P_1 \subseteq PP(x_1, \dots, x_n)$ be the set of all irreducible factors of elements of P . Set $P_2 := P_1 \setminus \{x_1, \dots, x_n\}$ and $P_3 := P \setminus P_1$.
- 4: **for all** $p \in P_2$ **do**
- 5: If $p \in P$, set $\beta_p := \sigma_p$, else set $\beta_p := \text{ALOP}$.
- 6: Set $\bar{\beta}_p := \beta_p \wedge \text{PolynomialSign}(p; \alpha_1, \dots, \alpha_n)$
- 7: If $\bar{\beta}_p = \text{NOOP}$, set *unsat* := *true* and return Q, τ, unsat
- 8: **end for**
- 9: For each $p \in P_3$, set $\gamma_p := \sigma_p$.
- 10: **repeat**
- 11: set *changed* := *false*.
- 12: for each $p \in P_1$, if $p \in P_2$, set $\delta_p := \beta_p$, else set $\delta_p := \alpha_i$, where $p = x_i$.
- 13: set $(\bar{P}_3; \bar{\delta}; \bar{\gamma}; \text{unsat}) := \text{BlackBox}(P_1; P_3; \delta; \gamma)$
- 14: if *unsat* = *true* return Q, τ, unsat
- 15: **if** $\bar{P}_3 \neq P_3$ or $\bar{\gamma} \neq \gamma$ or $\bar{\delta} \neq \delta$ **then**
- 16: set *changed* := *true*
- 17: set $P_3 := \bar{P}_3$ and $\gamma := \bar{\gamma}$.
- 18: for each $p \in P_1$, if $p \in P_2$, set $\bar{\beta}_p := \bar{\delta}_p$, else set $\bar{\alpha}_i := \bar{\delta}_p$, where $p = x_i$.
- 19: **end if**
- 20: set $(\bar{\alpha}_1, \dots, \bar{\alpha}_n), \bar{\beta}, \text{unsat} := \text{WhiteBox}(P_2; P_3; \alpha_1, \dots, \alpha_n; \beta; \gamma)$
- 21: if *unsat* = *true* return Q, τ, unsat
- 22: **if** $(\bar{\alpha}_1, \dots, \bar{\alpha}_n) \neq (\alpha_1, \dots, \alpha_n)$ or $\bar{\beta} \neq \beta$ **then**
- 23: set $(\alpha_1, \dots, \alpha_n) := (\bar{\alpha}_1, \dots, \bar{\alpha}_n)$ and $\beta := \bar{\beta}$.
- 24: set *changed* := *true*.
- 25: **end if**
- 26: **until** *changed* = *false*
- 27: **for all** $p \in P_2$ **do**
- 28: If $\beta_p = \text{NEOP}$, p is a factor of $q \in P_3$ and $\gamma_q \in \{LTOP, GTOP, NEOP\}$, set $\beta_p := \text{ALOP}$
- 29: If $\beta_p \neq \text{ALOP} \wedge \beta_p = \text{PolynomialSign}(p; \alpha_1, \dots, \alpha_n)$ set $\beta_p := \text{ALOP}$.
- 30: If $\beta_p \neq \text{ALOP}$, let $R := (P_2 \setminus \{p\}) \cup P_3$ and $\rho := (\beta \mid P_2 \setminus \{p\}) \cup \gamma$. If $\beta_p = \text{DeduceAll}(p; R; \alpha_1, \dots, \alpha_n; \rho)$ set $\beta_p := \text{ALOP}$.
- 31: **end for**
- 32: **for all** $p \in P_3$ **do**
- 33: if $\gamma_p \neq \text{ALOP} \wedge \gamma_p = \text{PolynomialSign}(p; \alpha_1, \dots, \alpha_n)$, set $\gamma_p := \text{ALOP}$
- 34: **end for**
- 35: Set $Q_1 := \{x_i : \alpha_i \neq \text{ALOP}\}$, $Q_2 := \{p \in P_2 : \beta_p \neq \text{ALOP}\}$ and $Q_3 := \{p \in P_3 : \gamma_p \neq \text{ALOP}\}$.
- 36: Set $Q := Q_1 \cup Q_2 \cup Q_3$, set $\tau_{x_i} := \alpha_i$ for each $x_i \in Q_1$, $\tau_p := \beta_p$ for each $p \in Q_2$, and $\tau_p := \gamma_p$ for each $p \in Q_3$
- 37: return Q, τ, unsat

easily seen to terminate. Assuming the elements of the input P are presented in factored form so that P_1 is explicit in the input, all other loops in Simplify (i.e. other than the repeat/until) iterate over subsets of the input. Combin-

ing this with our earlier assertion that sub-algorithms PolynomialSign, DeduceAll, BlackBox and WhiteBox all run in polynomial time shows that the running time of Simplify is indeed polynomial in the input size.

6. INTEGRATION WITH CAD AND QE BY VIRTUAL TERM SUBSTITUTION

We used the inequality simplification algorithm to improve performance of two key algorithms used in solving problems related to systems of polynomial equations and inequalities over the reals: cylindrical algebraic decomposition (CAD), and quantifier elimination by virtual term substitution.

Applying simplification to QE by virtual term substitution is straightforward — simplify prior to eliminating any quantified variables, and simplify after each variable-elimination step. Our simplification is used in addition to a method based on [12], and our empirical comparisons are with virtual term substitution using only the method based on [12].

CAD construction is a big topic, and giving an overview is well beyond what can be accomplished here. However, those familiar with CAD will recognize that *projection* and *lifting* (or *stack construction*) are the principal phases of CAD construction. We use simplification in three ways. First, the input is simplified before CAD construction is attempted. Second, in the projection phase, we use simplification to reduce the size of the projection set. Third, in the lifting phase, we use deduced inequalities and equations to truncate stack construction.

6.1 Reducing projection set size in CAD

There are several different general-purpose projection operators for CAD construction: Collins’ original projection [9], Hong’s projection [13], McCallum’s projection [15], and the McCallum-Brown projection [2]. Each presents many different ways in which simplification could be used to reduce projection sets. Our use is fairly straightforward. Let F be the input formula.

With the McCallum-Brown projection, which is used for “well-oriented” inputs, for each projection factor p with main variable x , for each coefficient q of $p(x)$, if $F \wedge q$ simplifies to false, then it suffices to include q in the projection set in lieu of considering the system defined by the vanishing of all coefficients of $p(x)$.

With Hong’s projection, which is used for inputs that are not “well-oriented”, for each projection factor $p = p_n x^n + \dots + p_1 x + p_0$, one normally adds all coefficients to the projection factor set. If $F \wedge p_i$ simplifies to false, then it suffices to include $p_n, \dots, p_i$ in the projection set and not $p_{i-1}, \dots, p_0$. Hong’s projection also includes the sequence of principal subresultant coefficients of projection factors in the projection set. If $F \wedge psc_i$ simplifies to false, it suffices to include the principal subresultant coefficients only up to the i th in the projection set.

6.2 Truncating lifting steps in CAD

The lifting phase of CAD construction constructs an explicit data structure representing a CAD of $\mathbb{R}^n$. This data structure is a tree, in which nodes of depth i correspond to cells in a CAD of $\mathbb{R}^i$. In [10], Collins and Hong introduced “partial” CAD construction, in which construction of some branches of this tree structure is truncated based on evaluating the input formula F at “sample points”. Our simplification algorithm gives another test that can determine

that this truncation is valid. For each projection factor p , we apply Algorithm Simplify to $F \wedge p$ and obtain a sign-condition that p must satisfy. If any of these conditions are violated at a sample point, stack construction at that point can be truncated.

7. EXPERIMENTAL RESULTS

The inequality simplification algorithm as well as the CAD algorithm have been implemented in C, as a part of the kernel of *Mathematica*. The virtual substitution algorithm has been implemented in the *Mathematica* programming language. The experiments have been conducted on a 2.8 GHz Intel Xeon processor, with 72 GB of RAM available.

To measure the performance improvement of the CAD algorithm we needed a large collection of “naturally occurring” CAD inputs. To obtain such collection of examples we run benchmark tests for *Mathematica* equation and inequality solving and optimization functions and captured CAD inputs containing at least three variables. This way we collected 2498 distinct CAD inputs. Then we selected those inputs for which the timing of at least one of the methods (with or without simplification) was between 50 milliseconds and 5 minutes. The obtained collection contains 209 examples.

For the virtual substitution we used two collections of examples. The first one contains 16 examples derived from various applications. The second one contains 40 randomly generated examples. Unlike for the CAD algorithm, we can use randomly generated examples here, since the virtual substitution algorithm itself generates simplifiable systems of inequalities.

Results of the experiments are summarized in Table 1. Pairs of columns marked CAD, VSA and VSR give comparison results for, respectively, CAD examples, virtual substitution examples from applications and randomly generated virtual substitution examples. Columns marked Y and N give the results for algorithms, respectively, using and not using our simplification procedure. For each example, let T denote the computation time. For the CAD algorithm, let S denote the number of cells constructed in the lifting phase. For the virtual substitution algorithm, let S denote the total number of equations and inequalities appearing in the result, where each $f\sigma$, for $\sigma \in \{<, \leq, >, \geq, =, \neq\}$, is counted with a weight equal to the number of factors of f . Let subscripts Y and N indicate whether the algorithm uses our simplification procedure. The row marked “# examples” gives the total number of examples. The row marked “-20% time” gives the number of examples for which, in column Y, $T_Y < 0.8T_N$, in column N, $T_N < 0.8T_Y$. The row marked “-20% size” gives the number of examples for which, in column Y, $S_Y < 0.8S_N$, in column N, $S_N < 0.8S_Y$. The row marked “Max time factor” gives the maximum values of, in column Y, T_N/T_Y , in column N, T_Y/T_N . The row marked “Max size factor” gives the maximum values of, in column Y, S_N/S_Y , in column N, S_Y/S_N . Rows marked “Mean T_N/T_Y ” and “Mean S_N/S_Y ” give the values of the geometric mean of T_N/T_Y and S_N/S_Y . In virtual substitution examples from applications, in one of the examples the algorithm using simplification was able to eliminate one quantifier more than the algorithm not using simplification. In another example the algorithm using simplification took 81 milliseconds and the algorithm not using simplification did not finish in an hour, hence the example is not included in the table.

Table 1: Experimental results

Simplification	CAD		VSA		VSR	
	Y	N	Y	N	Y	N
# examples	209		15		40	
-20% time	74	0	6	6	0	33
-20% size	77	1	11	0	9	0
Max time factor	2199	1.17	106	2.33	1.18	3.77
Max size factor	2397	1.31	74811	1	41948	1.01
Mean T_N/T_Y	1.48		2.26		0.615	
Mean S_N/S_Y	1.82		19.0		1.60	

8. CONCLUSIONS

The goal of this work was “fast simplification” — a simplification algorithm that is fast enough to be applied within other algorithms that compute with Tarski formulas without ever slowing them down significantly, but which also produces useful simplification in a substantial number of cases. The empirical results from the previous section bear out that, in CAD construction and in using virtual term substitution for problems arising from applications, our simplification algorithm is able to dramatically improve the time required or the quality of the result produced in some instances, without ever having any serious negative impact. When applying virtual term substitution to randomly generated problems, the improvement is not as clear. This is not terribly surprising, because examples arising in practice tend to be very non-random.

There are two major avenues for moving forward from this work. The first is to look at more kinds of white-box deductions. This could allow for deductions to be made in more circumstances, although presumably at the cost of increased running time on all or inputs. The second is a more in depth study of how simplification or the deductions made during simplification can improve CAD construction. Certainly what we present in this paper is not an exhaustive list of possibilities.

9. REFERENCES

- [1] ARNON, D. S., COLLINS, G. E., AND MCCALLUM, S. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal on Computing* 13, 4 (1984), 865–877.
- [2] BROWN, C. W. Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation* 32, 5 (November 2001), 447–465.
- [3] BROWN, C. W. Simple CAD construction and its applications. *Journal of Symbolic Computation* 31, 5 (May 2001), 521–547.
- [4] BROWN, C. W. Fast simplifications for Tarski formulas. In *ISSAC '09: Proceedings of the 2009 international symposium on Symbolic and algebraic computation* (New York, NY, USA, 2009), ACM, pp. 63–70.
- [5] BROWN, C. W. Algorithm MinWtBasis for simplifying conjunctions of monomial inequalities. Tech. Rep. USNA-CS-TR-2010-01, U.S. Naval Academy Computer Science Department, 2010.
- [6] BROWN, C. W., AND DAVENPORT, J. H. The complexity of quantifier elimination and cylindrical algebraic decomposition. In *ISSAC '07: Proceedings of the 2007 international symposium on Symbolic and algebraic computation* (New York, NY, USA, 2007), ACM, pp. 54–60.
- [7] BUCHFUHRER, D., AND UMANS, C. The complexity of boolean formula minimization. In *Proceedings of Automata, Languages and Programming, 35th International Colloquium* (2008), pp. 24–35.
- [8] CAVINESS, B., AND JOHNSON, J. R., Eds. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Texts and Monographs in Symbolic Computation. Springer-Verlag, 1998.
- [9] COLLINS, G. E. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Lecture Notes In Computer Science* (1975), vol. 33, Springer-Verlag, Berlin, pp. 134–183. Reprinted in [8].
- [10] COLLINS, G. E., AND HONG, H. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation* 12, 3 (Sep 1991), 299–328.
- [11] DAVENPORT, J. H., AND HEINTZ, J. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation* 5 (1997), 29–35.
- [12] DOLZMANN, A., AND STURM, T. Simplification of quantifier-free formulae over ordered fields. *Journal of Symbolic Computation* 24, 2 (Aug. 1997), 209–231. Special Issue on Applications of Quantifier Elimination.
- [13] HONG, H. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proc. International Symposium on Symbolic and Algebraic Computation* (1990), pp. 261–264.
- [14] HONG, H. Simple solution formula construction in cylindrical algebraic decomposition based quantifier elimination. In *ISSAC '92: Papers from the international symposium on Symbolic and algebraic computation* (New York, NY, USA, 1992), ACM, pp. 177–188.
- [15] MCCALLUM, S. An improved projection operator for cylindrical algebraic decomposition. In *Quantifier Elimination and Cylindrical Algebraic Decomposition* (1998), B. Caviness and J. Johnson, Eds., Texts and Monographs in Symbolic Computation, Springer-Verlag, Vienna.
- [16] WEISPFENNING, V. The complexity of linear problems in fields. *J. Symb. Comput.* 5, 1-2 (1988), 3–27.
- [17] WEISPFENNING, V. Quantifier elimination for real algebra — the quadratic case and beyond. *AAECC* 8 (1997), 85–101.
- [18] YANAMI, H., AND ANAI, H. Development of synrac—formula description and new functions. In *Proceedings of International Workshop on Computer Algebra Systems and their Applications (CASA) 2004* (2004), vol. 3039 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 286–294.

Parametric Quantified SAT Solving

Thomas Sturm

Departamento de Matemáticas, Estadística y
Computación, Universidad de Cantabria
39071 Santander, Spain
sturmt@unican.es

Christoph Zengler

Symbolic Computation Group
Wilhelm-Schickard-Institut, Universität Tübingen
72076 Tübingen, Germany
zengler@informatik.uni-tuebingen.de

ABSTRACT

We generalize successful algorithmic ideas for quantified satisfiability solving to the parametric case where there are parameters in the input problem. The output is then not necessarily a truth value but more generally a propositional formula in the parameters of the input. Since one can naturally embed propositional logic into first-order logic over Boolean algebras, our work amounts from a model-theoretic point of view to a quantifier elimination procedure for initial Boolean algebras. Our work is completely and efficiently implemented in the logic package Redlog contained in the open source computer algebra system Reduce. We describe this implementation and discuss computation examples pointing at possible applications of our work to configuration problems in the automotive industry.

Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*Computations on Discrete Structures*; F.4.1 [Mathematical Logic and Formal Languages]: Mathematical Logic—*Computational Logic*; G.4 [Mathematical Software]: Algorithm design and analysis; I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms

General Terms

Algorithms, Performance, Theory

Keywords

Propositional Logic, SAT, QSAT, Parameters, Generalization, Quantifier Elimination

1. INTRODUCTION

Satisfiability solving (SAT) and quantified satisfiability solving (QSAT) for propositional logic are canonical complete problems for the complexity classes NP and PSPACE,

respectively [5, 10]. Especially in the last 15 years there has been considerable research in these areas [16, 17, 27, 8]. This is motivated on the one hand by the increasing number of practical applications like bounded model checking [4, 3]. On the other hand, finding efficient heuristics for these two problems provides via polynomial reduction efficient algorithms for all problems in NP and PSPACE, respectively. Furthermore algorithmic ideas from SAT and QSAT have been successfully transferred to dedicated algorithms for e.g., graph coloring or constraint satisfaction problems.

In an earlier paper [21], the first author and others have discussed how to formally integrate SAT and QSAT into their first-order computer logic system Redlog [7]. Their approach is essentially to consider first-order formulas in the language of Boolean algebras over the theory of initial Boolean algebras, i.e., Boolean algebras freely generated by the empty set. Then first-order formulas can be brought into a normal form, where every atomic formula is of the form $v = 0$ or $v = 1$ for variables v . This way, quantifier-free formulas can be directly interpreted and presented to the user as propositional formulas. SAT then amounts to deciding the existential closure of a given formula, and QSAT corresponds to the decision of an arbitrary first-order sentence.

Decision procedures for first-order theories have historically often been in fact quantifier elimination procedures, even long before this term was formally introduced. Prominent examples are Presburger's completeness proof for the additive theory of the integers or Tarski's decision procedure for real closed fields [19, 24]. Based on the virtual substitution approach [26] it was straightforward to devise in [21] a quantifier elimination procedure for initial Boolean algebras. Since variable-free atomic formulas are obviously decidable this quantifier elimination procedure covers in particular both SAT and QSAT. Additionally, it admits to consider formulas, where only some variables are quantified while others remain free and are considered parameters of the problem. One then obtains via quantifier elimination a quantifier-free formula exclusively in the parameters that is equivalent to the input formula. We refer to this procedure as *parametric quantified satisfiability solving* (PQSAT). Elsewhere this has been referred to as *open QBF* [1].

The asymptotic worst-case time complexity of the procedure in [21] is bounded by a single exponential function in the input length, where the input problem needs not be in any Boolean normal form. It is not hard to see that this bound is tight for the considered problem. However, the existing successful research on SAT and QSAT teaches us that such crude complexity considerations are not sufficient to draw conclusions about the practical applicability of an al-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

gorithm. In fact, from the benchmarks in [21] it is quite clear that this work could not compete with state-of-the-art SAT or QSAT checkers. From a SAT solving point of view the procedure applied to sentences was roughly an implementation of the classical DLL algorithm [6] without learning and without non-chronological backtracking facilities.

In the present paper, we are now going to generalize to PQSAT more sophisticated and practically successful approaches to SAT and QSAT solving, viz. DLL with conflict driven clause learning and non-chronological backtracking. We have implemented our algorithm PQSAT including the underlying QSAT solver in the logic package Redlog contained in the open-source computer algebra system Reduce¹. Our PQSAT uses QSAT mostly as a black box, which can easily be replaced by alternative QSAT solvers. On the basis of our description here, existing QSAT solvers can be extended to PQSAT in such a way that their performance on regular QSAT problems is not affected.

The plan of this paper as follows: In Section 2 we summarize the design and the properties of our underlying implementations of SAT and QSAT. Basic definitions and concepts introduced there are going to be reused for the description of PQSAT (and its specialization PSAT) in Section 3. We prove correctness and termination of our procedure and give some upper bounds for the asymptotic worst-case complexity. Section 4 presents an application example in the area of product configuration in the automotive industry and analyzes the performance of our methods and implementations by means comprehensive benchmarks taken from the literature as well as from industrial cooperation projects. Section 5 finally points at some future research directions.

2. REVISION OF SAT AND QSAT

In this section we are going to summarize the design and the properties of DLL SAT solvers and QSAT solvers to the extent necessary to describe in the following section our extensions of this approach to the parametric case. We also make clear, which design decisions we have taken for the implementation of our own underlying QSAT solver.

2.1 SAT

There has been considerable research on stochastic local search algorithms for SAT solving [22, 20]. This led to considerable improvements (1.324^n instead of 2^n) of the upper bound for the asymptotic worst-case complexity [12]. While these probabilistic algorithms perform very well on random input, the vast majority of SAT solvers successfully applied to real-world problems uses the *Davis–Logemann–Loveland (DLL) approach* [6]. All the winners of the last years’ SAT Races fall into this category including RSat [18], MiniSAT [8], and PicoSAT [2]. Since we are ourselves interested in practical applicability at the first place, we are going to focus on DLL here.

DLL is basically a complete search in the search space of all 2^n variable assignments with early cuts in the search tree when an unsatisfiable branch is detected. Based on this approach Silva and Sakallah have introduced a concept referred to as *clause learning* [16]. Their approach extends the classical DLL approach by *non-chronological backtracking* and *automatic learning of new clauses* in case of conflicts. Therefore this approach is referred to as *conflict-driven clause*

learning (CDCL). All successful solvers mentioned above are in fact CDCL solvers. CDCL solvers are CNF-only solvers, meaning the input formula must be converted into CNF before solving.

We use the standard notation of propositional logic with propositional variables from a suitable infinite set $\mathcal{V}$, Boolean operators $\neg, \vee, \wedge$, and Boolean constants true and false. An *assignment* is a partial function $\alpha : \mathcal{V} \rightarrow \{\top, \perp\}$ mapping variables to truth values. We write $x \leftarrow b$ for $\alpha(x) = b$, we denote by $\text{dom}(\alpha) \subseteq \mathcal{V}$ the variables that have been assigned, and we follow the convention to write $\alpha \models \varphi$ when some formula φ holds with respect to α . We write $\text{vars}(\varphi)$ to denote the finite set of variables occurring in a formula φ . A *conjunctive normal form* (CNF) is a conjunction of clauses. A *clause* is a disjunction $(\lambda_1 \vee \dots \vee \lambda_n)$ of literals. Each *literal* λ_i is either a variable $x_i \in \mathcal{V}$ or its logical negation $\neg x_i$. It is convenient to identify a CNF with the set of all clauses contained in it. An *empty clause* is a clause where all x_i have been assigned truth values in such a way that all corresponding λ_i evaluate to false and therefore the whole clause is false. It is obvious that once reaching an empty clause, no extension of the corresponding assignment can satisfy the CNF formula containing that clause. We call the occurrence of an empty clause a *conflict*. A *unit clause* is a clause where all but one x_i have been assigned, all λ_j for $j \neq i$ evaluate to false, and therefore in order to satisfy the clause the remaining *unit variable* x_i must be assigned such that λ_i becomes $\top$. The process of detecting unit clauses and fixing the corresponding values for the unit variables is called *unit propagation*, which plays an important role in modern SAT solvers.

On each *decision level* CDCL assigns variables and performs unit propagation until either an empty clause arises or the formula is satisfied. If the formula is satisfied CDCL returns true. In the case of an empty clause a resolution-based learning process is started at the end of which CDCL learns a new clause and backtracks to a certain decision level. If an empty clause arises at level 0, CDCL returns false. There are various strategies how to learn the new clause in the conflict case. In Redlog we use the firstUIP strategy described in [17]. Since modern SAT solvers spend up to 90% of their time performing unit propagation it is crucial to implement this efficiently. In Redlog we use the common concept of *watched literals* [17]. The idea is to observe two literals in each clause. As long as these two have no assigned truth value, the clause cannot be unit. When one of the literals gets assigned and evaluates to false, another unassigned literal is chosen to be guarded instead. If there is no other literal, the clause is unit, and we can perform unit propagation. The last important adjusting screw is the heuristics how to choose the next variable to be assigned. Our implementation offers various choices of selection heuristics. The default is a variation of the MOM heuristic [9], which prefers literals with a maximum occurrence in clauses of minimal size. MOM turned out to perform very well on our benchmarks discussed in Subsection 4.2.

2.2 QSAT

While SAT implicitly assumes all variables to be existentially quantified, QSAT more generally expects for each variable an explicit quantification, either existential or universal. We assume here that formulas are in *prenex normal form*, where all quantifiers precede a CNF. Whenever using non-

¹<http://reduce-algebra.sourceforge.net>

CNF formulas, we consider this an abbreviated notation for some equivalent formula in CNF. For detailed definitions of empty and unit clauses in the context of QSAT, see [27].

In the SAT case backtracking is only necessary when an empty clause is detected. For QSAT, in contrast, backtracking has to be performed possibly also in the case that one branch of the search tree is satisfiable: For each universally quantified variable x_i both branches, $x_i \leftarrow \top$ and $x_i \leftarrow \perp$, must be satisfiable. Therefore after assigning $x_i \leftarrow \perp$, there is a backtrack performed assigning $x_i \leftarrow \top$. We give the CDCL Algorithm for QSAT [27, 15, 11], which reflects the backtracking for universally quantified variables in lines 11–15.

Input: (φ, α) , where φ is a fully quantified propositional formula, and α is an optional assignment for variables existentially quantified in the outermost block of φ , $\emptyset$ by default

Output: $(\tau, \varphi', \alpha')$, where $\tau \in \{\text{true}, \text{false}\}$, $\varphi' \longleftrightarrow \varphi$ with additional learned clauses, and α' the final variable assignment

```

1 label all bindings in  $\alpha$  with level  $-1$ 
2 level := 0
3 while true do
4   unitPropagation()
5   if  $\varphi$  has an empty clause then
6     level := analyseConflict()
7     if level = 0 then
8       return (false,  $\varphi$ ,  $\alpha$ )
9     backtrack(level)
10  else
11    if  $\alpha \models \varphi$  then
12      level := analyseSAT()
13      if level = 0 then
14        return (true,  $\varphi$ ,  $\alpha$ )
15      backtrack(level)
16    else
17      level := level + 1
18      choose  $x \in \text{vars}(\varphi) \setminus \text{dom}(\alpha)$  wrt. to the
        quantification level
19       $\alpha := \alpha \cup \{(x \leftarrow \perp, \text{level})\}$ 

```

Algorithm 1: The QSAT algorithm $\text{QSAT}(\varphi, \alpha)$

The procedure `analyseConflict()` learns a new clause and returns a suitable non-negative backtrack level. During the procedure `analyseSAT()` the value of the last universally quantified variable is flipped in order to search both branches in the search tree. Notice that when replacing lines 12–15 by “return true” we obtain the CDCL SAT algorithm as described in Subsection 2.1. In fact if there are no universally quantified variables then QSAT proceeds exactly like CDCL.

For the variable selection in line 18 there are essentially the same heuristics used as with SAT. There is, however, one important restriction: Successive quantifiers of the same type are grouped like

$$Q_1 x_1 \dots Q_1 x_{k_1} Q_2 x_{k_1+1} \dots Q_2 x_{k_2} \dots \varphi,$$

where $Q_i \in \{\exists, \forall\}$, $Q_{i+1} \neq Q_i$, and $x_j \in \text{vars}(\varphi)$. The index $i \in \mathbb{N}$ of Q_i is the *quantification level* of the corresponding quantified variables $x_{k_{i-1}+1}, \dots, x_{k_i}$. The variable selection heuristics must choose a variable from the smallest quantifi-

cation level where there are still unassigned variables. Notice that in the worst case like $\exists x_1 \forall x_2 \exists x_3 \forall x_4 \dots \varphi$ one must successively pick $x_1, x_2, x_3, x_4, \dots$, i.e., there is no choice at all.

It is noteworthy that in contrast to most existing implementations of QSAT we do not only return τ but also φ' and α . This additional information is required for the PQSAT algorithm described in the next section. We have verified for several existing QSAT solvers that the code can be easily adapted to meet our requirements. Similarly existing solvers can easily be adapted to accept the additional input parameter α , in fact some already do so.

3. PARAMETRIC QSAT

PQSAT generalizes QSAT by admitting *free*, i.e. non-quantified, variables. It is important to understand that these free variables are, *not* assumed to be implicitly existentially quantified but *parameters* of the described problem. Consequently, PQSAT does in general not *decide* its input problem by returning true or false. Instead the output is a *disjunctive normal form (DNF)* establishing necessary and sufficient conditions on the free variables for the existence of a satisfying assignment. In the special case that for all possible assignments of the free variables the corresponding instances of QSAT yield $\tau = \text{true}$, PQSAT will return true as well. Analogously, PQSAT yields false if all QSAT instances return $\tau = \text{false}$.

Algorithm 2 states the general PQSAT algorithm. For an input formula φ with free variables, we split the set $\text{vars}(\varphi)$ of all variables into a set $\text{bvars}(\varphi)$ of bound (quantified) variables and a set $\text{fvars}(\varphi)$ of free variables.

Input: (φ, α) , where φ is a quantified propositional formula possibly containing free variables, and α is an optional partial assignment for $\text{fvars}(\varphi)$, $\emptyset$ by default

Output: (τ, φ') , where τ is a quantifier-free formula with $\text{vars}(\tau) = \text{fvars}(\varphi)$, and $\varphi' \longleftrightarrow \varphi$ with additional learned clauses

```

1 if  $\text{fvars}(\varphi) \setminus \text{dom}(\alpha) = \emptyset$  then
2    $(\sigma, \varphi', \beta) := \text{QSAT}(\exists \varphi, \alpha)$ 
3   if  $\sigma = \text{true}$  then
4     return (form( $\alpha$ ),  $\varphi'$ )
5   else
6     return (false,  $\varphi'$ )
7 else
8    $x := \text{choose a variable from } \text{fvars}(\varphi) \setminus \text{dom}(\alpha)$ 
9    $\alpha' := \alpha \cup \{x \leftarrow \perp\}$ 
10   $\psi_1 := \text{false}$ 
11  if no conflict is reached after unit propagation then
12     $(\psi_1, \varphi) := \text{PQSAT}(\varphi, \alpha')$ 
13   $\alpha'' := \alpha \cup \{x \leftarrow \top\}$ 
14   $\psi_2 := \text{false}$ 
15  if no conflict is reached after unit propagation then
16     $(\psi_2, \varphi) := \text{PQSAT}(\varphi, \alpha'')$ 
17  return (simplify( $\psi_1 \vee \psi_2$ ),  $\varphi$ )

```

Algorithm 2: The PQSAT algorithm $\text{PQSAT}(\varphi, \alpha)$

In the degenerate case that there are no free variables at all, our PQSAT algorithm will reduce to one call to QSAT. Recall from the previous section that the QSAT algorithm

in turn reduces to a CDCL SAT algorithm in the more degenerate case of a purely existential problem.

The algorithm is recursive in both its input parameters. It is noteworthy that for the initial call α can be used to fix in advance the assignment for some subset of free variables when experimenting with a problem. Notice that the return value α of QSAT is not essentially used here; it will be in our optimization for PSAT discussed in Subsection 3.3

The main idea of PQSAT is to use essentially the classical DLL algorithm for the free variables. Whenever in that course all free variables have been assigned, we have got a QSAT subproblem, for which we call QSAT(φ) and obtain either $(\text{true}, \varphi', \alpha)$ or $(\text{false}, \varphi', \alpha)$.

In line 2 we construct the existential closure of φ in order to meet the specification of QSAT. The existential quantifiers introduced this way are actually semantically irrelevant as all corresponding variables are already assigned by α .

Observe in line 12 that we save in φ the original input formula augmented by clauses additionally learned during the first recursive PQSAT call. This is propagated in line 16 to the second recursive PQSAT call. This leads to the effect that we *transport* learned clauses from one QSAT call to the next and thus avoiding repeatedly arriving at the same conflicts. It is not hard to see that since learning happens via resolution and resolution is compatible with substitution of truth values for variables the learned clauses in fact remain valid. The idea is visualized in Figure 3. In order to limit the blow up of φ in that course we use *activity heuristics* after each run of QSAT to delete learned clauses that have not significantly produced new conflicts in the past.

If ψ_1 or ψ_2 is false in line 17 then simplify $(\psi_1 \vee \psi_2)$ eliminates one superfluous false.

When QSAT returns (true, φ) we add a Boolean representation $\text{form}(\alpha)$ of the current variable assignment α to the output formula and proceed with the algorithm. Following the usual convention that empty conjunctions are true this Boolean representation is defined as

$$\text{form}(\alpha) = \bigwedge_{\substack{v \in \text{dom}(\alpha) \\ v(\alpha) = \top}} v \wedge \bigwedge_{\substack{v \in \text{dom}(\alpha) \\ v(\alpha) = \perp}} \neg v.$$

As an example consider

$$\text{form}(\{x \leftarrow \top, y \leftarrow \perp, z \leftarrow \perp\}) = x \wedge \neg y \wedge \neg z.$$

It is easy to see that for any assignments α, α' we have $\text{form}(\alpha) = \text{form}(\alpha')$ iff $\alpha = \alpha'$. Furthermore:

LEMMA 1 (UNIVERSAL PROPERTY). *Let α be an assignment. Then the following hold:*

- (i) $\text{vars}(\text{form}(\alpha)) = \text{dom}(\alpha)$ and $\alpha \models \text{form}(\alpha)$
- (ii) If γ is a conjunction of literals and $\text{vars}(\gamma) = \text{dom}(\alpha)$ and $\alpha \models \gamma$, then $\gamma = \text{form}(\alpha)$ up to commutativity. $\square$

Consider α with $\text{dom}(\alpha) = \text{fvars}(\varphi)$. Then up to commutativity $\text{form}(\alpha)$ is the unique conjunction of literals with $\text{vars}(\text{form}(\alpha)) = \text{fvars}(\varphi)$ and $\alpha \models \text{form}(\alpha)$.

Similar to DLL after each assignment of a free variable in line 9 or 13 we use unit propagation with watched literals to propagate the current assignment. If we encounter an empty clause, we cut the search tree.

To conclude this subsection Figure 3 visualizes a computation of PQSAT with $\varphi = \exists x \forall y ((x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w))$ and $\alpha = \emptyset$. Since QSAT yields true for the assignments

$$\{u \leftarrow \perp, w \leftarrow \perp\}, \quad \{u \leftarrow \perp, w \leftarrow \top\}, \quad \{u \leftarrow \top, w \leftarrow \top\}.$$

one finally obtains $\tau = (\neg u \wedge \neg w) \vee (\neg u \wedge w) \vee (u \wedge w)$.

3.1 Correctness and Termination

In this section we assume the correctness and termination of the CDCL QSAT procedure as proved in [27].

LEMMA 2. *Let (τ, φ') be the return value of some PQSAT call. Then τ is in DNF.*

PROOF. If $|\text{fvars}(\varphi)| = 0$, then $\tau = \text{form}(\alpha)$. By definition this is either true or a conjunction of literals, both of which are in DNF. For $|\text{fvars}(\varphi)| = n + 1$ we obtain essentially $\tau = \psi_1 \vee \psi_2$ with $|\text{fvars}(\psi_1)| = |\text{fvars}(\psi_2)| = n$. According to the induction hypothesis both ψ_1 and ψ_2 are in DNF and so is $\psi_1 \vee \psi_2$. $\square$

LEMMA 3. *Let φ be a quantified formula. Consider*

$$A = \{ \alpha \mid \text{fvars}(\varphi) = \text{dom}(\alpha), \text{QSAT}(\varphi, \alpha) = (\text{true}, \varphi', \alpha') \}.$$

Then $\varphi \longleftrightarrow \bigvee_{\alpha \in A} \text{form}(\alpha)$.

PROOF. To start with, observe that for each $\alpha \in A$ we have $\text{fvars}(\text{form}(\alpha)) = \text{dom}(\alpha) = \text{fvars}(\varphi)$ and thus

$$\text{fvars}\left(\bigvee_{\alpha \in A} \text{form}(\alpha)\right) = \text{fvars}(\varphi).$$

Let α_0 be an assignment with $\text{dom}(\alpha_0) = \text{fvars}(\varphi)$. Assume that $\alpha_0 \models \varphi$. Then $\text{QSAT}(\varphi, \alpha_0) = (\text{true}, \varphi', \alpha')$ for some φ', α' . It follows that $\alpha_0 \in A$. Since $\alpha_0 \models \text{form}(\alpha_0)$ we obtain $\alpha_0 \models \bigvee_{\alpha \in A} \text{form}(\alpha)$. Assume, vice versa, that $\alpha_0 \models \bigvee_{\alpha \in A} \text{form}(\alpha)$. Then $\alpha_0 \models \text{form}(\alpha_1)$ for some $\alpha_1 \in A$. It follows that

$$\text{vars}(\text{form}(\alpha_1)) = \text{dom}(\alpha_1) = \text{fvars}(\varphi) = \text{dom}(\alpha_0).$$

By Lemma 1(ii) we obtain $\text{form}(\alpha_1) = \text{form}(\alpha_0)$, which in turn implies $\alpha_1 = \alpha_0$. On the other hand we know $\text{QSAT}(\varphi, \alpha_1) = (\text{true}, \varphi', \alpha')$ for some φ', α' , and using the correctness of QSAT it follows that $\alpha_0 = \alpha_1 \models \varphi$. $\square$

THEOREM 1 (CORRECTNESS OF PQSAT). *Let φ be a quantified propositional formula and $(\tau, \varphi') = \text{PQSAT}(\varphi, \emptyset)$. Then τ is quantifier-free and $\tau \longleftrightarrow \varphi$.*

PROOF. By inspection of the algorithm we see that possible return values for τ are $\text{form}(\alpha)$ (line 4) or false (line 6) or disjunctions of these (line 17), all of which are quantifier-free.

Consider the special case that for all assignments α with $\text{dom}(\alpha) = \text{fvars}(\varphi)$ there has been QSAT(φ, α) called in line 2. Then it is easy to see that $\tau = \bigvee_{\alpha \in A} \text{form}(\alpha)$ as described in Lemma 3. Hence $\tau \longleftrightarrow \varphi$ by Lemma 3.

Consider now a particular assignment α_0 with $\text{dom}(\alpha_0) = \text{fvars}(\varphi)$ for which there has not been QSAT(φ, α_0) called. According to lines 11 and 14 of PQSAT then there exists a partial assignment $\alpha'_0 \subseteq \alpha_0$ causing a conflict, that is $\alpha'_0 \models \varphi \longleftrightarrow \text{false}$. It follows that $\alpha_0 \models \varphi \longleftrightarrow \text{false}$ and accordingly QSAT(φ, α_0) = $(\text{false}, \varphi', \alpha')$. Hence that missing QSAT call is irrelevant for the semantics of τ . $\square$

THEOREM 2 (TERMINATION OF PQSAT). *PQSAT terminates.*

PROOF. We have to show that there is no infinite recursion. Since $|\text{fvars}(\varphi) \setminus \text{dom}(\alpha)| \in \mathbb{N}$ decreases with every recursive call either in line 12 or line 16 due to the assignments in line 9 or line 13, respectively, the condition $\text{fvars}(\varphi) \setminus \text{dom}(\alpha) = \emptyset$ in line 1 finally becomes true, and the algorithm returns in line 4 or line 6. $\square$

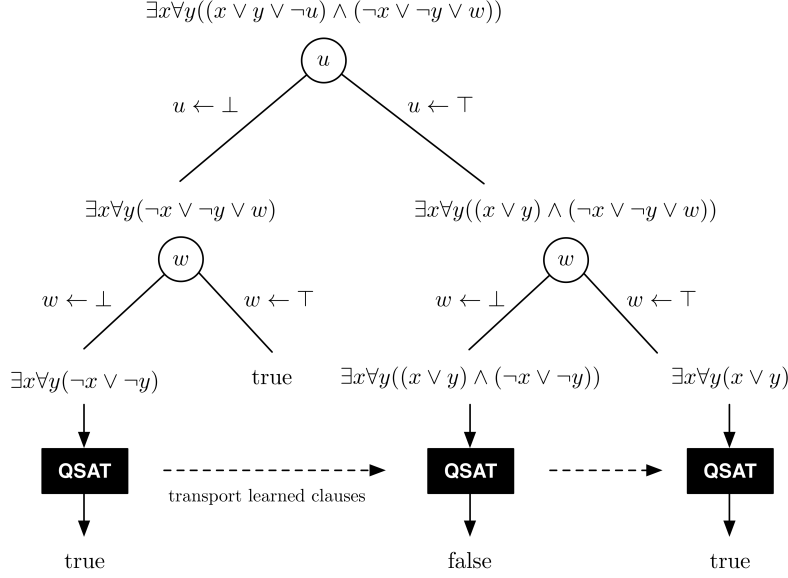


Figure 1: Example PQSAT computation

3.2 Complexity

THEOREM 3 (COMPLEXITY OF PQSAT). *Consider as complexity parameters $f = |\text{fvars}(\varphi)|$ and $b = |\text{bvars}(\varphi)|$. Then the asymptotic time complexity of PQSAT is bounded by 2^{f+b} in the worst case. In particular this complexity is bounded by $2^{\text{length}(\varphi)}$.*

PROOF. Consider an input formula φ and let f and b be as above. QSAT is obviously bounded by 2^b . In PQSAT the QSAT algorithm is called at most 2^f times. We hence obtain $2^f \cdot 2^b = 2^{f+b}$. $\square$

3.3 PSAT and Local Search

We consider the special case that the input formula of PQSAT does not contain any universal quantifier but possibly existential quantifiers and free variables. We are going to refer to such formulas as existential formulas. Naturally we refer to PQSAT for existential formulas as PSAT. For PSAT problems we use ideas from probabilistic SAT [22] to improve our PQSAT algorithm. The key idea is the following: When we have found a satisfying assignment we may expect that there are further satisfying assignments “close” to the found one. We are now going to make precise this idea.

The *Hamming distance* between two assignments is defined as the number of variables for which the assignments differ:

$$d(\alpha, \alpha') = |\{x \in \text{dom}(\alpha) \mid \alpha(x) \neq \alpha'(x)\}|.$$

The *Hamming circle* $H(\alpha, r)$ with center α and radius r is the set of all assignments α' with $d(\alpha, \alpha') \leq r$.

Let φ be an existential formula without universal quantifiers. Denote by $\hat{\varphi}$ the *matrix* of φ , i.e. the formula without quantifiers. Let α be an assignment with $\text{dom}(\alpha) = \text{fvars}(\varphi)$, and β be an assignment with $\text{dom}(\beta) \subseteq \text{bvars}(\varphi)$ such that $\alpha \cup \beta \models \hat{\varphi}$. It is going to turn out *a posteriori* that for our industrial application problems discussed in Subsection 4.1

there is a significant probability that there exist further $\alpha' \in H(\alpha, r)$ with $r = 25$ such that $\alpha' \cup \beta \models \hat{\varphi}$ as well. We have determined that number $r = 25$ heuristically.

We are now going to optimize our PQSAT algorithm to *locally search* the Hamming circle whenever finding a satisfying assignment. In the successful cases where $\alpha' \in H(\alpha, r)$ with $\alpha' \cup \beta \models \hat{\varphi}$ this saves expensive calls to $\text{QSAT}(\varphi, \alpha')$. In the unsuccessful cases, however, we cannot draw any conclusions: If for an assignment β we have $\alpha' \cup \beta \models \hat{\varphi} \iff \text{false}$, then there can be a different β' with $\alpha' \cup \beta' \models \hat{\varphi}$.

Since assignments are now checked at two different places, viz. local search and calls to QSAT, we maintain a set s of hashes of already checked assignments α in order to avoid duplicate checking.

Algorithm 3 states the “if” part of the PSAT algorithm. Input, output and the “else” part are literally as in Algorithm 2 (lines 8–17).

```

1 if fvars( $\varphi$ ) \setminus \text{dom}(\alpha) = \emptyset then
2   if hash( $\alpha$ )  $\notin s$  then
3     ( $\sigma, \varphi', \beta$ ) := QSAT( $\exists \varphi, \alpha$ )
4      $s := s \cup \{\text{hash}(\alpha)\}$ 
5     if  $\sigma = \text{true}$  then
6        $\psi_1 := \text{form}(\alpha)$ 
7       ( $\psi_2, s$ ) := localSearch( $\hat{\varphi}, \alpha, r, \beta, s$ )
8       return (simplify( $\psi_1 \vee \psi_2$ ),  $\varphi'$ )
9     else
10      return (false,  $\varphi'$ )
11 else
12   return (false,  $\varphi$ )

```

Algorithm 3: The relevant part of PSAT(φ, α)

Note that in contrast to PQSAT we cannot use α to fix an assignment in advance. This would require some slight modifications. The procedure `localSearch()` used there is

going to be given as Algorithm 4.

Input: $(\hat{\varphi}, \alpha, r, \beta, s)$ where $\hat{\varphi}$ is a quantifier-free formula, $\alpha \cup \beta$ is an assignment with $\text{dom}(\alpha \cup \beta) \subseteq \text{vars}(\hat{\varphi})$, $r \in \mathbb{N}$, and s is a set of hashes

Output: (ψ, s') where ψ is a quantifier-free formula with $\text{vars}(\psi) = \text{dom}(\alpha)$, and s' is a set of hashes

```

1  $\psi := \text{false}$ 
2 foreach  $\alpha' \in H(\alpha, r)$  do
3   if  $\alpha' \cup \beta \models \hat{\varphi}$  then
4      $\psi := \psi \vee \text{form}(\alpha')$ 
5      $s := s \cup \{\text{hash}(\alpha')\}$ 
6 return  $(\psi, s)$ 

```

Algorithm 4: $\text{localSearch}(\hat{\varphi}, \alpha, r, \beta, s)$

To conclude this section we discuss why these ideas cannot be straightforwardly generalized to PQSAT. Recall that the starting point of our search is a satisfying assignment $\alpha \cup \beta \models \hat{\varphi}$. The role of β is to serve as a witness for the satisfiability wrt. α of the corresponding existentially quantified formula φ . Since in the general case there are possibly universally quantified variables such a witness cannot exist for principle reasons.

4. APPLICATIONS AND BENCHMARKS

Besides the application examples for PQSAT mentioned in [21] we are going to present an application for PSAT originating from a cooperation with the automotive industry. We will describe this application in Subsection 4.1 before we go on to benchmarks and comparisons to other systems in Subsection 4.2.

4.1 Configurations in the Automotive Industry

In the automotive industry, the compilation and maintenance of correct product configuration data is a complex task. In [23, 14] there is described how valid configurations of constructible vehicles can be expressed using quantifier-free propositional formulas. We are now going to present a simplified version of this method in order to explain our new application which is based on such descriptions.

For many positions in a vehicle one can choose between various parts to fill that position. Each part p , e.g. engine, steering wheel, or left mirror, of a vehicle is mapped to an equipment code x_p . Many parts correspond to *customer options*. It is important to understand however that there are thousands of parts, the vast majority of which is not directly selected by the customer. For our discussion here we refer to those parts as *hidden parts*. The set of all constructible vehicles is described by one formula referred to as *product overview formula* (POF). A POF is a conjunction of rules. A single rule is either a *constructibility condition* (CC) or a *supplementary code* (SC). A CC is an implication $x_p \rightarrow \varphi$ where φ is an arbitrary quantifier-free propositional formula. It must hold when $x_p \leftarrow \top$. An SC is an implication $\varphi \rightarrow x_p$ which must hold when a certain condition φ holds.

Consider as a toy example a vehicle where the customer options are three different engines with equipment codes e_1 , e_2 , e_3 , and three additional features a_1 , a_2 , a_3 . As hidden parts we consider two different gearboxes g_1 , g_2 . There is exactly one engine in a vehicle (cc_1 – cc_4) and exactly one

gearbox (cc_5 – cc_7):

$$\begin{aligned}
\text{cc}_1 &= \text{true} \rightarrow e_1 \vee e_2 \vee e_3, & \text{cc}_5 &= \text{true} \rightarrow g_1 \vee g_2, \\
\text{cc}_2 &= e_1 \rightarrow \neg e_2 \wedge \neg e_3, & \text{cc}_6 &= g_1 \rightarrow \neg g_2, \\
\text{cc}_3 &= e_2 \rightarrow \neg e_1 \wedge \neg e_3, & \text{cc}_7 &= g_2 \rightarrow \neg g_1. \\
\text{cc}_4 &= e_3 \rightarrow \neg e_1 \wedge \neg e_2,
\end{aligned}$$

Engine e_1 must be combined with gearbox g_1 , e_2 must be combined with g_2 , and e_3 can be combined with g_1 or g_2 :

$$\text{cc}_8 = e_1 \rightarrow g_1, \quad \text{cc}_9 = e_2 \rightarrow g_2, \quad \text{cc}_{10} = e_3 \rightarrow g_1 \vee g_2.$$

Feature a_2 must not be combined with a_3 , the combination of e_3 and g_1 must be combined with a_2 , and the combination of e_3 and g_2 must be combined with a_3 :

$$\text{cc}_{11} = a_2 \rightarrow \neg a_3, \quad \text{sc}_1 = e_3 \wedge g_1 \rightarrow a_2, \quad \text{sc}_2 = e_3 \wedge g_2 \rightarrow a_3.$$

The POF is the conjunction of all CCs and SCs:

$$\text{POF} = \bigwedge_{i=1}^{11} \text{cc}_i \wedge \bigwedge_{j=1}^2 \text{sc}_j.$$

When a customer chooses a certain option p , the currently used configuration tool adds $x_p \leftarrow \top$ to an assignment α . For each customer option p' that is not chosen it adds $x_{p'} \leftarrow \perp$. At the end there runs an automatic assignment process, which iteratively adds $x_q \leftarrow \top$ to α for all SCs $\varphi \rightarrow x_q$ with $\alpha \models \varphi$. For hidden parts $x_{q'}$ it also adds $x_{q'} \leftarrow \perp$ for all SCs $\varphi \rightarrow \neg x_{q'}$ with $\alpha \models \varphi$. Notice that customer options can be flipped from $\perp$ to $\top$ but not vice versa. A car is considered constructible if and only if its POF is satisfiable wrt. to the final α . In the positive case α encodes the configuration of the car. Notice that this configuration cannot be obtained straightforwardly by pure SAT solving.

The problem with the automatic assignment process is that it does not necessarily terminate and strongly depends on the order of adding assignments. Furthermore in a significant number of cases it delivers false negatives, i.e. turns the POF unsatisfiable via α although the vehicle is constructible in reality.

Our solution is to use PSAT as a less efficient but more powerful fallback option. In the case that the POF has turned out unsatisfiable for some order we proceed as follows: We start with the original input α of the automatic assignment process. For each assignment $x_p \leftarrow \top$ we conjunctively add x_p to the POF. Then we delete *all* assignments from α . Notice that we completely ignore assignments $x_p' \leftarrow \perp$ corresponding to customer options. Finally all codes corresponding to customer options are considered as free variables, while all codes corresponding to hidden parts are existentially quantified. The result of PSAT is then $\bigvee_i \tau_i$ where each conjunction τ_i of literals describes one possible way to render the vehicle constructible by specifying the absence and presence of customers options that were not mentioned in the original order.

We continue the example above. Assume that a customer chooses e_3 and a_1 . We compute

$$(\tau_1 \vee \tau_2, \varphi') = \text{PSAT}(\exists g_1 \exists g_2 (\text{POF} \wedge e_3 \wedge a_1), \emptyset),$$

where

$$\tau_1 = \neg e_2 \wedge \neg e_1 \wedge a_3 \wedge \neg a_2, \quad \tau_2 = \neg e_2 \wedge \neg e_1 \wedge \neg a_3 \wedge a_2.$$

For each τ_i we are interested in the subset of positive literals, which specify additionally required customer options. These

subsets are presented to the user as the final output:

$$\{\{a_2\}, \{a_3\}\}.$$

We see that the either code a_2 or code a_3 must be chosen, but not both. Our final output can be processed by a human expert. Alternatively one can automatically select subsets by, e.g., minimizing the number of necessary changes or costs.

Our benchmarks discussed in the next section will demonstrate that our approach and our implementation of PSAT in Redlog is capable of solving such problems on real instances from current product lines of vehicles with thousands of variables and ten thousands of clauses.

4.2 Benchmarks

To start with, we would like to point out that all examples discussed so far take less than 10 ms CPU time, which corresponds to the accuracy of the timing facilities built into Reduce on our architecture. We have used PSL-based Reduce on an Apple Mac Pro with two 2.8 GHz Quad-Core Intel Xeon Processors using one core of one processor and 750 MB of memory.

Table 1 shows computations with three instances of configuration problems in the automotive industry as described in the previous subsection. We have taken these instances from the publicly available benchmark suite of DaimlerChrysler’s Mercedes car lines.² The names of the instances in the table are followed by number of variables and number of clauses. We compare the computation times of PQSAT and PSAT for an increasing number of free variables. For PSAT we also give the *l.s. rate*, which is the percentage of QSAT calls saved by local search. One can clearly see that for all examples PSAT outperforms PQSAT. Up to 75% of calls to QSAT can be saved, and we observe significant speed up factors, in particular with many free variables.

In Table 2 we compare PQSAT with QE [21]. QE is implemented in Reduce as well such that the computation times are absolutely comparable. We use a set of standard benchmarks for SAT solvers and QSAT solvers. We restrict to quite small examples such that also QE finishes within reasonable time. We consider three SAT benchmarks:

ii8 stems from the Boolean formulation of a problem of inductive interference [13].

sinz is a superset of the formulas considered in Table 1.²

auto is a set of product configuration formulas as described in Subsection 4.1. It is currently used in industry.

For QSAT we consider two benchmarks:

toilet is the *bomb in the toilet* planning problem [25].

2player has been introduced and discussed in [21].

The last section of Table 2 shows the results of our PQSAT benchmarks. In order to get large PQSAT benchmarks we have deleted some quantifiers from the *bomb in the toilet* [25] problem.

It is noteworthy that on these PQSAT benchmarks the performance of QE, in contrast to that of PQSAT, increases when increasing the number of free variables in a fixed formula. This observation is compatible with the complexity

²http://www-sr.informatik.uni-tuebingen.de/~sinz/DC/DC_base.zip

analysis in [21]: QE is single exponential in the number of quantifiers but only polynomial in all other reasonable complexity parameters. Recall that our PQSAT, in contrast, is single exponential in *all* variables. For all benchmarks considered here PQSAT clearly outperforms QE.

Finally we would like to remark that our current implementation of SAT and QSAT cannot compete with the current highly specialized solvers [17, 8, 18, 2]. Recall, however, that our approach is mostly generic and compatible with these solvers.

5. FUTURE WORK

Since PQSAT uses the general idea of the DLL approach our next research goal is to adapt more of the recent developments in SAT solving to our algorithm. This includes learning for free variables. There are interesting research perspectives concerning various heuristics used throughout this paper, e.g. the Hamming radius for the local search. Concerning applications our configuration technique discussed in Subsection 4.1 can of course be transferred to other product lines. A not so obvious but probably very interesting example is software configuration. As additional application areas we are considering to study bounded model checking and software verification.

6. REFERENCES

- [1] M. Benedetti and H. Mangassarian. QBF-based formal verification: Experience and perspectives. *JSAT*, 5:133–191, 2008.
- [2] A. Biere. Picosat essentials. *JSAT*, 4:75–97, 2008.
- [3] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. In M. Zelkowitz, editor, *Highly Dependable Software*, volume 58 of *Advances in Computers*. Academic Press, San Diego, CA, 2003.
- [4] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Form. Methods Syst. Des.*, 19(1):7–34, 2001.
- [5] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the STOC ’71*, pages 151–158. ACM Press, New York, NY, 1971.
- [6] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.
- [7] A. Dolzmann and T. Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, 1997.
- [8] N. Eén and N. Sörensson. An extensible SAT-solver. In *SAT*, volume 2919 of *LNCS*, pages 502–518. Springer, 2003.
- [9] J. W. Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1995.
- [10] H. Fujiwara and S. Toida. The complexity of fault detection problems for combinational logic circuits. *IEEE Trans. Comput.*, 31(6):555–560, 1982.
- [11] E. Giunchiglia, M. Narizzano, and A. Tacchella. Learning for quantified boolean logic satisfiability. In *Eighteenth National Conference on Artificial Intelligence*, pages 649–654. American Association for Artificial Intelligence, Menlo Park, CA, 2002.

Table 1: Comparison of PQSAT and PSAT (all times in s)

free variables	C168_FW (1909/7477)			C129_FR(1888/7404)			C211_FW(1665/5929)		
	PQSAT	PSAT	l.s. rate	PQSAT	PSAT	l.s. rate	PQSAT	PSAT	l.s. rate
0	2.3	2.3	0%	2.8	2.8	0%	1.4	1.4	0%
5	2.6	2.4	0%	3.5	3.6	0%	1.9	1.9	0%
10	2.4	2.5	25%	4.8	4.1	74%	2.3	2.3	0%
15	3.1	2.5	25%	7.0	4.7	27%	2.3	2.3	6%
20	6.1	3.1	14%	17.0	6.4	30%	2.4	2.5	21%
25	11.5	3.2	10%	111.0	37.8	27%	4.1	2.4	65%
30	281.0	45.9	9%	—	—	—	8.3	3.5	56%
35	—	—	—	—	—	—	9.9	4.9	26%
40	—	—	—	—	—	—	50.5	24.5	30%

Table 2: Benchmark for PQSAT and the quantifier elimination procedure (QE) from [21]

Benchmark	instances	variables	free variables	clauses	QE time in s	PQSAT time in s
ii8	41	66–1068	0	186–821	3230.00	8.50
sinz	36	1411–1909	0	1982–11342	4227.00	99.67
auto	8	2291–4223	0	3006–16387	16650.00	61.20
toilet_a_02	5	18–90	0	39–408	310.00	< 0.01
toilet_a_04	9	32–140	0	129–894	4780.00	< 0.01
2player ($n = 250$)	1	500	0	998	5.10	0.03
2player ($n = 500$)	1	1000	0	1998	36.00	0.08
2player ($n = 750$)	1	1500	0	2998	115.90	0.18
toilet_a_04_01.4	2	60	20, 40	229	2.40	3.00
toilet_a_06_01.4	3	86	20, 40, 60	649	121.30	12.90
toilet_a_08_01.2	2	60	20, 40	2205	21.30	8.10

- [12] K. Iwama and S. Tamaki. Improved upper bounds for 3-SAT. In *Proceedings of the SODA '04*, pages 328–328. SIAM, Philadelphia, PA, 2004.
- [13] A. P. Kamath, N. K. Karmarkar, K. G. Ramakrishnan, and M. G. C. Resende. A continuous approach to inductive inference. *Mathematical Programming*, 57(1-3):215–238, 1992.
- [14] W. K  chlin and C. Sinz. Proving consistency assertions for automotive product data management. *J. Autom. Reasoning*, 24(1-2):145–163, 2000.
- [15] R. Letz. Lemma and model caching in decision procedures for quantified boolean formulas. In *Proceedings of the TABLEAUX '02*, pages 160–175. Springer, 2002.
- [16] J. P. Marques-Silva, K. A. Sakallah, J. P. Marques, S. Karem, and A. Sakallah. Conflict analysis in search algorithms for propositional satisfiability. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, 1996.
- [17] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: engineering an efficient SAT solver. In *Proceedings of the DAC '01*, pages 530–535. ACM, New York, NY, 2001.
- [18] K. Pipatsrisawat and A. Darwiche. Rsat 2.0: SAT solver description. Technical Report D-153, Computer Science Department, UCLA, 2007.
- [19] M. Presburger.   ber die Vollst  ndigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du premier congr  s de Math  maticiens des Pays Slaves*, pages 92–101, Warsaw, Poland, 1929.
- [20] U. Sch  ning. New algorithms for k-SAT based on the local search principle. In *Proceedings of the MFCS '01*, pages 87–95. Springer, 2001.
- [21] A. M. Seidl and T. Sturm. Boolean quantification in a first-order context. In *Proceedings of the CASC 2003*, pages 329–345. Technische Universit  t M  nchen, Munich, Germany, 2003.
- [22] B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532, 1995.
- [23] C. Sinz, A. Kaiser, and W. K  chlin. Formal methods for the validation of automotive product configuration data. *Artif. Intell. Eng. Des. Anal. Manuf.*, 17(1):75–97, 2003.
- [24] A. Tarski. A decision method for elementary algebra and geometry. Prepared for publication by J. C. C. McKinsey. RAND Report R109, RAND, Santa Monica, CA, 1948.
- [25] R. Waldinger. The bomb in the toilet. *Computational Intelligence*, 3(1):220–221, 1987.
- [26] V. Weispfenning. The complexity of linear problems in fields. *J. Symbolic Computation*, 5(1-2):3–27, 1988.
- [27] L. Zhang and S. Malik. Conflict driven learning in a quantified boolean satisfiability solver. In *Proceedings of the ICCAD '02*, pages 442–449. ACM, New York, NY, 2002.

A Method for Semi-Rectifying Algebraic and Differential Systems using Scaling Type Lie Point Symmetries with Linear Algebra

François Lemaire
University of Lille I, LIFL
Villeneuve d'Ascq, France
Francois.Lemaire@lifl.fr

Aslı Ürgüplü
University of Lille I, LIFL
Villeneuve d'Ascq, France
Asli.Urguplu@lifl.fr

ABSTRACT

We present two new algorithms based on Lie symmetries that respectively allow to semi-rectify algebraic systems and reduce the number of parameters on which the steady points of a differential system depend. These algorithms facilitate the qualitative analysis of algebraic and differential systems. They are designed with a strong view towards applications, such as modeling in biology. Their implementation, already available in our MABSys package, is of polynomial time complexity in the input size.

Categories and Subject Descriptors

G.4 [Mathematics of Computing]: Mathematical Software—*Algorithm design and analysis*; I.6.3 [Computing Methodologies]: Simulation and Modeling—*Applications*; J.3 [Computer Applications]: Life and Medical Sciences

General Terms

Algorithms, Design, Theory

Keywords

Modeling, qualitative analysis, Lie point symmetries

1. INTRODUCTION

The Lie symmetry theory provides well-known tools for exact symbolic simplification of algebraic and differential systems (see [16, 12, 3, 4]). In this paper, we propose an exact simplification method based on new algorithms that use the classical Lie symmetry theory for medium size systems (about twenty coordinates). This method ensures the semi-rectification of algebraic systems and at the same time extend the classical reduction based on *scaling* type Lie symmetries of a differential system (see [12]) by semi-rectifying its steady points. Roughly speaking, the semi-rectification of an algebraic system consists in finding an explicit change of coordinates such that the solutions in the new coordinates

set form a (generalized) cylinder (i.e. some new coordinates are free). More technically, the change of coordinates for semi-rectifying an algebraic system is computed using some of its symmetries. Moreover, the semi-rectification of the steady points of a differential system Σ is obtained by applying on Σ the change of coordinates computed for semi-rectifying the steady points of Σ .

In this paper, we assume that some coordinates are positive, which is often the case when they describe physical amounts or parameters of parametric systems; for example in biology. Our change of coordinates has the nice property of keeping the positiveness of the positive coordinates in the new coordinates set, which can be an important knowledge (see [1]). Our algorithms were developed and are designed with a strong view towards applications, such as modeling in biology. We facilitate the qualitative analysis of parametric algebraic and differential systems (assumed to be continuous dynamical systems), that is certainly difficult because of the number of involved coordinates.

Indeed thanks to our algorithms, in the case of an algebraic system, the solutions depend on less coordinates meaning that some coordinates are made free. In the case of a differential system, the variety of the steady points depends on less parameters. As a consequence, after the change of coordinates, some parameters has no effect on the location of the steady points, which implies that they only have an effect on the dynamics of the system. Moreover, the system in the original coordinates and the one in the new coordinates are equivalent. This ensures that any qualitative result true in the new coordinates is also true in the original coordinates. This equivalence is guaranteed by the explicitly computed change of coordinates.

We restrict ourselves to a special family of change of coordinates: the monomial maps. This restriction allows us to have a global explicit change of coordinates, a strong condition which is difficult to ensure in general (usually change of coordinates are local and rarely explicit). Our algorithms are of polynomial time complexity in the input size thanks to the restriction of the set of Lie symmetries to *scalings* and some computational strategies such as the probabilistic resolution of systems (for the computation of these scalings). These are implemented in our MABSys package (see [9, 17]). Furthermore, we have chosen to be accessible to non-expert users in mathematics: the use of our algorithms does not require any knowledge about Lie symmetries.

Section 2 presents the problem we address. Section 3 and 4 respectively explain our method for algebraic and differential systems. Last section illustrates the interest of our meth-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

ods on a model of a genetic network involving a single self-regulated gene (see [2, 17]).

2. STATEMENT OF THE PROBLEM

We present two related algorithms that respectively handle algebraic and differential systems. In both cases, the goal is to simplify the systems study. In fact, we find an explicit change of coordinates such that the system rewritten in the new coordinates is easier to study in the following sense. In the case of an algebraic system, the solutions depend on less coordinates, meaning that some coordinates are made free. In the case of a differential system, the variety of the steady points depends on less parameters. Here are some examples (for which the complete computations are given in the next sections) to illustrate our algorithms.

EXAMPLE 2.1. Assume we are interested in solutions such that $a > 0$ and $b > 0$. Both systems

$$\begin{cases} by^2 - a = 0, \\ ax - b = 0 \end{cases} \quad \text{and} \quad \begin{cases} (\tilde{y}^2 - 1)\tilde{a}^2 = 0, \\ (\tilde{x} - 1)\tilde{b}^2 = 0 \end{cases} \quad (1)$$

are equivalent under the change of coordinates

$$a = \tilde{a}^2, \quad b = \tilde{b}^2, \quad x = \frac{\tilde{x}\tilde{b}^2}{\tilde{a}^2}, \quad y = \frac{\tilde{y}\tilde{a}}{\tilde{b}}. \quad (2)$$

Remark that the second system is easier to study since the coordinates $\tilde{a}$ and $\tilde{b}$ are now free (only $\tilde{x}$ and $\tilde{y}$ are constrained), whereas no coordinates were free in the first system.

EXAMPLE 2.2. Both differential systems (with $a, b > 0$)

$$\begin{cases} \dot{x} = by^2 - a, \\ \dot{y} = ax - b \end{cases} \quad \text{and} \quad \begin{cases} \dot{\tilde{x}} = (\tilde{y}^2 - 1)\frac{\tilde{a}^4}{\tilde{b}^2}, \\ \dot{\tilde{y}} = (\tilde{x} - 1)\frac{\tilde{b}^3}{\tilde{a}} \end{cases} \quad (3)$$

are equivalent under the change of coordinates (2) given in the previous example. In the new coordinates, the differential system still depends on the two parameters $\tilde{a}$ and $\tilde{b}$. However, its steady points expressions do not depend on $\tilde{a}$ and $\tilde{b}$. This improvement implies that $\tilde{a}$ and $\tilde{b}$ only have an effect on the nature of these steady points, not on their location.

2.1 Semi-Rectification

Our two algorithms are in fact based on the problem that we call the *semi-rectification of an algebraic system* and which is stated below.

In this paper, we work with polynomial algebraic systems defined by $S = \{s_1, \dots, s_t\}$ written in the coordinate set $Z = (z_1, \dots, z_n)$ with rational coefficients i.e. in $\mathbb{Q}[Z]$. In the sequel, we assume that the e first z_i 's are positive, where $1 \leq e \leq n$. The remaining coordinates are supposed either nonnegative, non-positive or arbitrary. Consequently, we only consider the solutions of S in the set

$$E = \{(z_1, \dots, z_n) \mid z_1 > 0, \dots, z_e > 0, z_{e+1} \in E_{e+1}, \dots, z_n \in E_n\} \quad (4)$$

where each E_i is either $\mathbb{R}^+$, $\mathbb{R}^-$ or $\mathbb{R}$.

The problem of semi-rectification of an algebraic system follows: given an algebraic system S , find an explicit diffeomorphism¹ Φ from E to E such that the system $\tilde{S}$, obtained by rewriting S in the new coordinates $\tilde{Z} = \Phi(Z)$, satisfies:

¹A diffeomorphism Φ from M to N is a bijection from M to N such that both Φ and Φ^{-1} are differentiable.

C1 the equations of $\tilde{S}$ are also polynomial,

C2 the solutions (in E) of $\tilde{S}$ can be described with equations involving less coordinates than S .

The motivation is that the system $\tilde{S}$ is easier to study because of **C2**. Moreover, restricting to diffeomorphisms from E to E ensures that the solutions (in E) of S and $\tilde{S}$ are in bijection. As a consequence, all the information between the original and the simplified systems is kept.

In this paper we restrict the family of change of coordinates to monomial maps of the form:

$$\tilde{z}_j = \Phi_j(Z) = \prod_{k=1}^n z_k^{C_{k,j}} \quad \forall j \in \{1, \dots, n\} \quad (5)$$

where the $C_{k,j}$'s are elements of a matrix C . Moreover, we impose the following conditions denoted by **(H)**:

1. C is an upper $n \times n$ invertible matrix with rational coefficients,
2. C^{-1} has only integer coefficients,
3. the block of the last $n - e$ lines of C is equal to the block of the last $n - e$ lines of the $n \times n$ identity matrix.

It is easy to show that these conditions ensure that the change of coordinates (5) is a C^∞ -diffeomorphism from E to E , and that the inverse change of coordinates Φ^{-1} of (5) (obtained by inverting the matrix C) only involves integer exponents (**C1** satisfied). **C2** is a straightforward consequence of lemma 3.3 and is detailed in section 3.2.

In order to solve this semi-rectification problem, we propose a method based on *scaling type Lie point symmetries*. The needed computations only require linear algebra over $\mathbb{Q}$ and involve what we call *semi-rectified algebraic systems*.

2.2 Semi-Rectified Algebraic Systems

DEFINITION 2.3. An algebraic system S is z_l -semi-rectified if $l \leq e$ and if for any solution $(z_1^0, \dots, z_n^0)$ of S in E , for any positive real $\bar{z}_l^0$, $(z_1^0, \dots, z_{l-1}^0, \bar{z}_l^0, z_{l+1}^0, \dots, z_n^0)$ is also a solution of S .

Roughly speaking, an algebraic system is z_l -semi-rectified if its solutions in E do not depend on z_l (or the coordinate z_l is free). Note that the condition $l \leq e$ is important and in the sequel, we only free some of the positive parameters, namely $\{z_1, \dots, z_e\}$. Remark that an algebraic system may be z_l -semi-rectified even if its equations involve the coordinate z_l . However, if a system S is z_l -semi-rectified, one can easily form a system S_l not involving z_l which has the same solutions as S using one of the following two lemmas, whose proofs are left to the reader.

LEMMA 2.4. Let S be a z_l -semi-rectified algebraic system and S_l defined by replacing z_l by the value 1 in S . Then the solutions of S and S_l , both taken in E , are the same.

LEMMA 2.5. Let $S = \{s_1, \dots, s_t\}$ be a z_l -semi-rectified algebraic system defined in the coordinates $Z = (z_1, \dots, z_n)$. For each polynomial s_i in S , write $s_i = \sum_{j=0}^{d_i} s_{ij} z_l^j$ where d_i is the degree of s_i in z_l and the s_{ij} are polynomials free of z_l . Then the solutions of S and $S_l = \{s_{ij}\}_{1 \leq i \leq t, 0 \leq j \leq d_i}$, both taken in E , are the same.

Once one has a z_i -semi-rectified algebraic system, the application of lemma 2.4 or lemma 2.5 ensures condition **C2** as shown in the following example.

EXAMPLE 2.6. *An easy computation shows that the algebraic system $S = \{(b - c^2)a + (c - d)a^2, (c - d)a^3\}$ is a -semi-rectified assuming $a > 0$. On the one hand lemma 2.4 yields the algebraic system $S_l = \{b - c^2 + c - d, c - d\}$ and on the other hand lemma 2.5 yields $S_l = \{b - c^2, c - d\}$. The two resulting systems do not involve the coordinate a anymore.*

2.3 Related Works

There exist widely used general strategies for simplification of algebraic and differential systems. The lumping, the sensitivity analysis or the time-scale analysis (see [11]), all decrease the number of coordinates but they cause a loss of information about individual original coordinates. In our work, we keep the explicit relationships between the original and the simplified systems, thanks to our explicit change of coordinates.

The dimensional analysis (see [7]) is a classical reduction method based on the units of coordinates. It simplifies large-scale problems using dimensionless parameters. In fact, the reduction of parameters of a differential system through the dimensional analysis is a special case of the reduction using scalings. This last method can be found in [6, 14] with an algorithm of polynomial time complexity in the input size. The method we propose in this paper is complementary to this reduction.

There exist many softwares on Lie symmetries. Some of them compute just Lie symmetries of algebraic and differential systems. Some others, like [5, 10], perform the reduction of systems using these symmetries. However, they do not deal with the *semi-rectification* that we present.

3. SEMI-RECTIFICATION OF ALGEBRAIC SYSTEMS

In this section, we first present some mathematical background about scaling type Lie symmetries needed to understand the algorithms we propose for the semi-rectification. Then we restate this problem in terms of scalings. Finally we show how to build a change of coordinates which solves our semi-rectification problem and we present the algorithm for the semi-rectification of algebraic systems.

3.1 Scaling type Lie Point Symmetries

Here, we define scaling type Lie symmetries of an algebraic system. We do not show their computation (see § 3.1.2).

3.1.1 Definitions

Roughly speaking a *Lie point symmetry* is a transformation that maps every solution of a system to another solution of the same system. In this paper, we consider *scaling* type Lie point symmetries of algebraic systems (see § 2.1 of [12]). More precisely, we consider invertible one-parameter ν transformation groups acting on $Z = (z_1, \dots, z_n)$ such that:

$$z_i \rightarrow z_i \nu^{\alpha_i} \quad \text{where} \quad i \in \{1, \dots, n\}, \alpha_i \in \mathbb{Q}, \nu \in \mathbb{R}_*^+. \quad (6)$$

In our algorithms, we use an infinitesimal approach (see [16]) for the computation of symmetries. Indeed, scalings can be represented by differential operators that act on the coordi-

nates Z of the studied system as follows:

$$\delta = \sum_{i=1}^n \alpha_i z_i \frac{\partial}{\partial z_i} \quad (7)$$

where the α_i 's are in $\mathbb{Q}$. In the sequel, the differential operator δ will simply be called a scaling.

EXAMPLE 3.1. *Let us illustrate the scalings on the following algebraic system:*

$$\begin{cases} by^2 - a &= 0, \\ ax - b &= 0 \end{cases} \quad (8)$$

written in the coordinate set $Z = (a, b, x, y)$. These two equations are invariant under the following invertible one-parameter ν transformation group with $\nu \in \mathbb{R}_*^+$:

$$a \rightarrow a\nu, \quad b \rightarrow b\nu, \quad x \rightarrow x, \quad y \rightarrow y. \quad (9)$$

If one applies (9) over any equation of (8), one gets the same equation, possibly up to some non-zero multiplicative factor. The transformation (9) is said to be a scaling and it can be represented by the following differential operator:

$$\delta = a \frac{\partial}{\partial a} + b \frac{\partial}{\partial b}. \quad (10)$$

DEFINITION 3.2. *A semi-rectified symmetry is a scaling that can be represented by a differential operator that acts on exactly one coordinate z i.e. by $\delta = \alpha z \partial / \partial z$ with α in $\mathbb{Q}$.*

Semi-rectified symmetries (see § 2.2 of [16] for *normal forms*) are particular cases of scalings such that all coefficients α_i are zero except one. They have a crucial role in our problem statement in terms of scalings because of the following lemma that ensures **C2**.

LEMMA 3.3. *Let S be an algebraic system defined on the set $Z = (z_1, \dots, z_n)$. If S possesses a semi-rectified symmetry represented by $\delta = \alpha_i z_i \partial / \partial z_i$ with $i \leq e$, then S is z_i -semi-rectified.*

PROOF. If δ is a scaling of S then its solutions in E are invariant under the following associated one-parameter ν group of transformation (see § 1.2 of [12]) with ν in $\mathbb{R}_*^+$:

$$\begin{cases} z_j &\rightarrow z_j \quad \forall j \neq i, \\ z_i &\rightarrow z_i \nu^{\alpha_i}. \end{cases} \quad (11)$$

This tells that if the point $Z^0 = (z_1^0, \dots, z_n^0)$ in E is a solution of S then the point $(z_1^0, \dots, z_{i-1}^0, \nu^{\alpha_i} z_i^0, z_{i+1}^0, \dots, z_n^0)$ is also a solution of S for ν in a neighborhood of 1. As S is an algebraic system, the last statement is also true for ν in $\mathbb{R}_*^+$, which implies that S is z_i -semi-rectified. $\square$

3.1.2 Computation and Implementation Remarks

The **MABSys** package, where the forthcoming semi-rectification algorithms are implemented, relies on the **ExpandedLiePointSymmetry** package (see [15]) for the computation of scaling type Lie point symmetries of algebraic systems. The complexity of the algorithm employed for this issue is polynomial in the input size (see proposition 4.2.8 in [17], [6, 14]). This gain of complexity arises mostly from the limitation to only scalings and the restriction of the general definition of Lie symmetries.

3.2 Semi-Rectification in terms of Scalings

The semi-rectification problem can be rewritten as follows. Given an algebraic system S whose solutions are taken in E , find a diffeomorphism from E to E such that the system $\tilde{S}$ (obtained by rewriting S in the new coordinates) admits as much semi-rectified symmetries as possible.

This problem solved by algorithm 2 relies on three steps:

1. to find a set of scalings associated to the algebraic system S given in the input,
2. to deduce a monomial map $\tilde{Z} = \Phi(Z)$ on E from these scalings,
3. to rewrite the algebraic system S in these new coordinates $\tilde{Z}$ in order to obtain $\tilde{S}$.

The first point is not treated in this paper (see § 3.1). The second point is the core of the semi-rectification and is performed by algorithm 1. By construction, the new coordinates we define by Φ corresponds to *invariants* and *semi-invariants* of the scalings computed at step 1. The third point necessitates a simple substitution of the new coordinates into the original system. As we will see in algorithm 1, the change of coordinates is computed in such a way that $\tilde{S}$ admits semi-rectified symmetries, which ensures that $\tilde{S}$ is semi-rectified for some coordinates because of lemma 3.3 (C2 satisfied).

REMARK 3.4. The method proposed in this section for the semi-rectification of algebraic systems follows the idea of the reduction method (see [12]) with minor differences. For example, our method preserves the number of coordinates as well as the positivity of the e first coordinates.

EXAMPLE 3.5. The algebraic system (8) has two scalings that can be represented by:

$$\delta_1 = -2a \frac{\partial}{\partial a} + 2x \frac{\partial}{\partial x} - y \frac{\partial}{\partial y}, \quad \delta_2 = -2b \frac{\partial}{\partial b} - 2x \frac{\partial}{\partial x} + y \frac{\partial}{\partial y}. \quad (12)$$

Assuming that a and b are positive coordinates, the semi-rectification process allows us to find an invertible change of coordinates $\tilde{Z} = \Phi(Z)$ with $\tilde{Z} = (\tilde{a}, \tilde{b}, \tilde{x}, \tilde{y})$ such that the scalings δ_1 and δ_2 are rewritten as:

$$\delta_1 = \alpha_{\tilde{a}} \tilde{a} \frac{\partial}{\partial \tilde{a}}, \quad \delta_2 = \alpha_{\tilde{b}} \tilde{b} \frac{\partial}{\partial \tilde{b}} \quad (13)$$

with $\alpha_{\tilde{a}}$ and $\alpha_{\tilde{b}}$ in $\mathbb{Q}$. Because of lemma 3.3, this implies that the system (8) is $\tilde{a}$ -semi-rectified and $\tilde{b}$ -semi-rectified in $\tilde{Z}$.

3.3 Computing the change of coordinates

In this section, we show how to compute our change of coordinates from a set of scalings, using only linear algebra.

3.3.1 Matrix of Scaling

In the forthcoming algorithms, the scalings of the studied system are handled all together thanks to the associated *matrix of scaling*. The left multiplication of such a matrix of scaling performs a linear combination of the associated scalings. Its right multiplication rewrites the associated scalings in a new coordinate set defined by the second factor.

DEFINITION 3.6. Let $\mathcal{B} = \{\delta_1, \dots, \delta_r\}$ be a set of scalings. For all i in $\{1, \dots, r\}$, denote $\delta_i = \sum_{j=1}^n \alpha_j^i z_j \partial / \partial z_j$ with α_j^i in $\mathbb{Q}$. The matrix of scaling associated to $\mathcal{B}$ is defined by:

$$M := \left(\alpha_j^i \right)_{1 \leq i \leq r, 1 \leq j \leq n}. \quad (14)$$

EXAMPLE 3.7. The matrix of scaling associated to two scalings given in (12) follows with $Z = (a, b, x, y)$:

$$M = \begin{pmatrix} -2 & 0 & 2 & -1 \\ 0 & -2 & -2 & 1 \end{pmatrix}. \quad (15)$$

3.3.2 Left and Right Multiplications

The following two lemmas clarify the left and the right multiplications of a matrix of scaling.

LEMMA 3.8. Let $\mathcal{B}_1 = \{\delta_1, \dots, \delta_r\}$ be a set of scalings acting on the coordinate set $Z = (z_1, \dots, z_n)$ and M the associated matrix of scaling of dimension $r \times n$. Let P be any invertible matrix of dimension $r \times r$ in $\mathbb{Q}$. Then the matrix PM is associated to a set of scalings $\mathcal{B}_2$ that generates the same vector space as $\mathcal{B}_1$. $\square$

PROOF. The scalings of $\mathcal{B}_1$ form a vector space thus performing left multiplication on M amounts to perform linear combination on the elements of $\mathcal{B}_1$. Since P is invertible, the vector space defined by $\mathcal{B}_1$ and that defined by $\mathcal{B}_2$ are the same. $\square$

LEMMA 3.9. Let $\mathcal{B} = \{\delta_1, \dots, \delta_r\}$ be a set of scalings of a system Σ defined on a coordinate set $Z = (z_1, \dots, z_n)$ with

$$\delta_i = \sum_{j=1}^n \alpha_j^i z_j \frac{\partial}{\partial z_j} \quad \forall i \in \{1, \dots, r\}, \alpha_j^i \in \mathbb{Q} \quad (16)$$

and M be associated matrix of scaling. Let C be a $n \times n$ invertible matrix, with coefficients in $\mathbb{Q}$, that defines a change of coordinates on Z as in (5). The matrix MC is a matrix of scaling of the system $\tilde{\Sigma}$ i.e. of the system Σ rewritten in $\tilde{Z}$. $\square$

PROOF. Let us denote by

$$\delta_i = \sum_{j=1}^n \tilde{\xi}_j^i \frac{\partial}{\partial \tilde{z}_j} \quad \forall i \in \{1, \dots, r\} \quad (17)$$

the scaling δ_i in the new coordinate set $\tilde{Z}$. One has (see ch. 1 of [13]):

$$\tilde{\xi}_j^i = \sum_{k=1}^n \alpha_k^i z_k \frac{\partial \tilde{z}_j}{\partial z_k} = \sum_{k=1}^n \alpha_k^i C_{k,j} \tilde{z}_k = \beta_j^i \tilde{z}_j. \quad (18)$$

By definition β_j^i are in $\mathbb{Q}$. So the coefficients of the scalings obtained from the change of coordinates (5) are given by MC and in addition they correspond to scalings of $\tilde{\Sigma}$. $\square$

In the following section we use these two multiplications when we deduce a change of coordinates from several scalings represented by a matrix of scaling.

3.3.3 Deducing a Change of Coordinates

In this section we present a way of deducing a monomial map from the scalings.

Deducing a Change of Coordinates from One Scaling.

In this paragraph, we show how to compute a change of coordinates from one scaling only, in order to give the idea of the algorithm 1. Let us write a scaling δ of an algebraic system S as follows:

$$\delta = \alpha_1 z_1 \frac{\partial}{\partial z_1} + \alpha_2 z_2 \frac{\partial}{\partial z_2} + \dots + \alpha_{n-1} z_{n-1} \frac{\partial}{\partial z_{n-1}} + \alpha_n z_n \frac{\partial}{\partial z_n}$$

but with the coefficients α_i in $\mathbb{Z}$.

REMARK 3.10. In practice, if the coefficients α_i are rational fractions then one can multiply the whole scaling by the lcm of their denominators. This multiplication by a constant in $\mathbb{N}$ does not modify the associated algebraic structure.

Suppose that $e = 1$ i.e. we consider solutions in E defined by $z_1 > 0$ and z_i is in E_i for $i \geq 2$. Assume that δ acts on the coordinate z_1 i.e. $\alpha_1 \neq 0$. To avoid rational powers, one first introduces a coordinate $\tilde{z}_1 = z_1^{1/\alpha_1}$ in case $\alpha_1 \neq 1$. Remark that $\tilde{z}_1$ verifies $\delta\tilde{z}_1 = \tilde{z}_1$. As a consequence, one obtains:

$$\delta = \tilde{z}_1 \frac{\partial}{\partial \tilde{z}_1} + \alpha_2 z_2 \frac{\partial}{\partial z_2} + \cdots + \alpha_{n-1} z_{n-1} \frac{\partial}{\partial z_{n-1}} + \alpha_n z_n \frac{\partial}{\partial z_n}.$$

Then, one can choose $n - 1$ supplementary new coordinates as follow:

$$\tilde{z}_1 = z_1^{1/\alpha_1}, \quad \tilde{z}_i = \frac{z_i}{z_1^{\alpha_i/\alpha_1}} \quad \forall i \in \{2, \dots, n\}. \quad (19)$$

where $\delta\tilde{z}_i = 0$ for all i in $\{2, \dots, n\}$ by construction. The monomial map (19) satisfies **(H)** and its inverse is simply

$$z_1 = \tilde{z}_1^{\alpha_1}, \quad z_i = \tilde{z}_i \tilde{z}_1^{\alpha_i} \quad \forall i \in \{2, \dots, n\}.$$

Consequently, in these new coordinates, the differential operator δ is a semi-rectified symmetry equal to $\tilde{z}_1 \partial / \partial \tilde{z}_1$, which ensures that $\tilde{S}$ is $\tilde{z}_1$ -semi-rectified.

EXAMPLE 3.11. Let us consider the scaling (10) of the algebraic system (8) defined on $Z = (a, b, x, y)$. Suppose that a is positive. According to (19), one defines a new coordinate set as follows:

$$\tilde{a} = a, \quad \tilde{b} = \frac{b}{a}, \quad \tilde{x} = x, \quad \tilde{y} = y. \quad (20)$$

The inverse of (20) gives the expressions to substitute into the system (8) in order to get the semi-rectified new system:

$$a = \tilde{a}, \quad b = \tilde{b}\tilde{a}, \quad x = \tilde{x}, \quad y = \tilde{y}. \quad (21)$$

Thus the new algebraic system writes:

$$\begin{cases} \tilde{b}\tilde{a}\tilde{y}^2 - \tilde{a} = 0, \\ \tilde{a}\tilde{x} - \tilde{b}\tilde{a} = 0, \end{cases} \Rightarrow \begin{cases} (\tilde{b}\tilde{y}^2 - 1)\tilde{a} = 0, \\ (\tilde{x} - \tilde{b})\tilde{a} = 0 \end{cases} \quad (22)$$

and it is $\tilde{a}$ -semi-rectified. Moreover, this new system possesses the semi-rectified symmetry represented by $\tilde{a}\partial/\partial\tilde{a}$.

This idea of deducing a change of coordinates is generalized in the following paragraph by an algorithm that takes into account several scalings of an algebraic system at the same time. Because one can compose transformation groups associated to scalings, it is possible to consider a set of scalings in our algorithms.

Deducing a Change of Coordinates from Several Scalings using Associated Matrix of Scaling.

In this paragraph, we present the algorithm 1 that permits to find a new coordinate set in which some scalings are transformed into semi-rectified symmetries.

PROOF OF ALGORITHM 1. The matrix of scaling M encodes the vector space generated by the scalings represented by the lines of M . One first computes the modified LU decomposition of $M = P L U_1 R$ where R is the $r \times n$ reduced row echelon form of M . By definition, the first non-zero entries in each row of R is equal to 1. Thanks to lemma 3.8,

Algorithm 1 GetChangeOfCoord(M, Θ, P)

Input: A matrix of scaling M of dimension $r \times n$ constructed w.r.t. the coordinate set $Z = (z_1, \dots, z_n)$.
A list of coordinates assumed positive $\Theta = (z_1, \dots, z_e)$.
A list of remaining coordinates $P = (z_{e+1}, \dots, z_n)$.

Output: A monomial map $\tilde{Z} = \Phi(Z)$.

A sublist $[\tilde{z}_{p_1}, \dots, \tilde{z}_{p_f}]$ of $\tilde{Z}$ with $f \leq \min(e, r)$ such that the vector space of the scalings of M rewritten in $\tilde{Z}$ contains the f semi-rectified symmetries $\tilde{z}_{p_i} \partial / \partial \tilde{z}_{p_i}$ with $1 \leq i \leq f$.

- 1: $R := \text{ReducedRowEchelonForm}(M)$;
- 2: *#Removing unnecessary symmetries*
- 3: $Q \leftarrow$ the matrix obtained from R by selecting the lines having at least one non-zero element in the first e columns;
- 4: $f \leftarrow$ the number of lines of Q ;
- 5: $[p_1, \dots, p_f] \leftarrow$ the list of column indices of the first non-zero elements in each line of Q ;
- 6: *#Construction of the inverse of the matrix C that encodes new coordinates.*
- 7: $C^{-1} \leftarrow$ the identity matrix of size $n \times n$;
- 8: $\forall i \leq f$, replace the line p_i of C^{-1} by the line i of Q ;
- 9: multiply each line of C^{-1} by the lcm of the denominators of the entries of the line;
- 10: **return** $\left[\text{seq}\left(\tilde{z}_j = \prod_{k=1}^n z_k^{C_{k,j}}, j = 1..n\right), [\tilde{z}_{p_1}, \dots, \tilde{z}_{p_f}] \right]$;

the matrices M and R represent the same vector space of scalings.

In order to build a monomial map satisfying the condition **(H)**, one gets rid of the scalings of R that do not act on Θ . Indeed, those removed scalings would otherwise introduce terms of the form z_i^α with $i > e$ and $\alpha \neq 1$ which would prevent the monomial map from being a diffeomorphism from E to E .

Removing these unnecessary scalings correspond to keeping the first f lines of R where f is the number of scalings that act at least on one positive coordinate. Let us denote this matrix by Q .

Thanks to lemma 3.9, finding our monomial map can be done by finding an invertible $n \times n$ matrix C that satisfies $QC = D$ where D is the matrix of scaling of dimension $f \times n$ associated to semi-rectified scalings that we are looking for. It is defined by $d_{ij} \neq 0$ if $i = p_j$ and 0 otherwise. The building of the inverse matrix C^{-1} satisfies such a property. Indeed after line 8, one has $Q = DC^{-1}$ with the non-zero entries in D being equal to 1. After line 9, the condition $Q = DC^{-1}$ can be kept by modifying the non-zero entries of D . Thus, the matrix C satisfies the condition **(H)**. $\square$

REMARK 3.12. The ordering of the coordinates in Θ is important. Indeed the reduced echelon form algorithm, which acts similarly to a Gaussian elimination, finds the pivots starting from the left. Then, the list of indices $[p_1, \dots, p_f]$ computed at line 5 is in fact the smallest list for the lexicographical order. This is important in practice: the coordinates for which one wants to semi-rectify a system should be listed by decreasing order of preference in the list Θ .

REMARK 3.13. Even by assuming that one has r independent scalings encoded in the matrix of scalings M , one can only obtain f semi-rectified symmetries with $f \leq r$. Indeed,

the point 3 in **(H)** is needed to have a diffeomorphism from E to E . This is why one has to get rid of the last lines of R at line 3.

EXAMPLE 3.14. Let us consider now the two scalings given in (12) and deduce a change of coordinates that semi-rectifies the algebraic system (8) using associated matrix of scaling M given in (15). We assume that a and b are positive coordinates. The unique row reduced echelon form

$$R = Q = \begin{pmatrix} 1 & 0 & -1 & \frac{1}{2} \\ 0 & 1 & 1 & -\frac{1}{2} \end{pmatrix} \quad (23)$$

of the matrix M represents the same vector space of scalings as M . In this case, the matrix Q is equal to R because the two scalings acts at least on a or b . Here are the matrix C^{-1} constructed using Q and its inverse C that encodes the new coordinates that we are looking for:

$$C^{-1} = \begin{pmatrix} 2 & 0 & -2 & 1 \\ 0 & 2 & 2 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad C = \begin{pmatrix} \frac{1}{2} & 0 & 1 & -\frac{1}{2} \\ 0 & \frac{1}{2} & -1 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (24)$$

The elements of C indicate the powers of the old coordinates in the new coordinates expressions. This change of coordinates (thus the new coordinate set) transforms the scalings represented by Q into semi-rectified symmetries given in (13) with $\alpha_{\tilde{a}} = \alpha_{\tilde{b}} = 2$. These scalings are represented by the following matrix of scaling:

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix} \quad (25)$$

written w.r.t. the new coordinates $\tilde{Z} = (\tilde{a}, \tilde{b}, \tilde{x}, \tilde{y})$. According to (5), one has:

$$\tilde{a} = a^{\frac{1}{2}}, \quad \tilde{b} = b^{\frac{1}{2}}, \quad \tilde{x} = \frac{x a}{b}, \quad \tilde{y} = \frac{y b^{\frac{1}{2}}}{a^{\frac{1}{2}}}. \quad (26)$$

The inverse of (26) follows:

$$a = \tilde{a}^2, \quad b = \tilde{b}^2, \quad x = \frac{\tilde{x} \tilde{b}^2}{\tilde{a}^2}, \quad y = \frac{\tilde{y} \tilde{a}}{\tilde{b}}. \quad (27)$$

One can rewrite the algebraic system (8) in these new coordinates $\tilde{Z}$ by substituting (27) into its equations. This procedure leads to the following algebraic system:

$$\begin{cases} (\tilde{y}^2 - 1) \tilde{a}^2 = 0, \\ (\tilde{x} - 1) \tilde{b}^2 = 0. \end{cases} \quad (28)$$

Remark that this system is at the same time $\tilde{a}$ -semi-rectified and $\tilde{b}$ -semi-rectified i.e. its positive solutions do not depend on the values of $\tilde{a}$ nor $\tilde{b}$.

3.4 Semi-Rectification Algorithm for Algebraic Systems

The algorithm 2 proceeds in three steps. The first step is the computation of the scalings of S at line 2 using the `ELPSymmetries` function of the `ExpandedLiePointSymmetry` package (see § 3.1). The second step builds the monomial map ensuring the conditions **C1** and **C2** from the scalings computed at line 2. The third step is simply the rewriting of S in the new coordinates. Since Φ satisfies **(H)**, line 7 yields an algebraic system. Line 8 ensures that $\tilde{S}$ is free of V .

Algorithm 2 SemiRectifyAlgebraicSystem(S, Θ, P)

Input: An algebraic system S written in $Z = (z_1, \dots, z_n)$.
A list of coordinates assumed positive $\Theta = (z_1, \dots, z_e)$.
A list of remaining coordinates $P = (z_{e+1}, \dots, z_n)$.
Output: A semi-rectified algebraic system $\tilde{S}$ that satisfies the conditions **C1** and **C2**.
A monomial map $\tilde{Z} = \Phi(Z)$ satisfying **(H)**.
A sublist V of $\tilde{Z}$ such that $\tilde{S}$ is semi-rectified for each element of V and $\tilde{S}$ is free of V .
1: # Computation of scalings
2: $Sym := \text{ELPSymmetries}(S, \text{sym}=\text{scaling})$;
3: # Computation of the change of coordinates
4: $M := \text{MatrixOfScaling}(Sym, Z)$;
5: $\tilde{Z} = \Phi(Z)$, $V := \text{GetChangeOfCoordinates}(M, \Theta, P)$;
6: # Computation of the semi-rectified system
7: $\tilde{S} \leftarrow \Phi^{-1}(S)$;
8: remove from $\tilde{S}$ the variables of V using lemma 2.5;
9: **return** $[\tilde{S}, \tilde{Z} = \Phi(Z), V]$;

REMARK 3.15. The line 2 of the algorithm 2 computes the scalings of S . Experimentally, we remarked that triangularizing S could help to find more scalings useful to simplify the solutions of the studied algebraic system. This is an option that uses the `RegularChains` package (see [8]) of Maple. The disadvantage is that the complexity of the associated computations is not polynomial in the worst case.

4. SEMI-RECTIFYING STEADY POINTS OF SYSTEMS OF ODES

The contribution of our semi-rectification method can be more easily observed on differential systems. The classical Lie symmetry theory provides tools to reduce the coordinates of a system of ODEs. For example, one can use scalings of the whole differential system to decrease its parameters number. We extend this simplification. The original idea of our semi-rectification procedure is to tackle the scalings of the algebraic system that defines the steady points of the studied system of ODEs. A scaling of the differential system is also a scaling of its steady points, but the converse is not true in general. However, we show that the scalings of the steady points can be used on the differential system to find a new coordinates set in which the variety of the steady points depends on less parameters. Moreover, the system in the original coordinates and the one in the new coordinates are equivalent, which ensures that any qualitative result (e.g. the absence of a Hopf bifurcation, see [1]) true in the new coordinates is also true in the original coordinates. The algorithm 3 semi-rectifies the steady points of a differential system.

In this paper, we consider parametric systems of ODEs of the form $\dot{Z} = F(Z)$ where $Z = (z_1, \dots, z_n)$ is a list of time depending functions. We encode the p parameters as constant functions (i.e. one asserts $\dot{z}_i = 0$ for all $i \leq p$). Thus one has $F(Z) = (0, \dots, 0, F_{p+1}(Z), \dots, F_n(Z))$ where each $F_i(Z)$ is an element of $\mathbb{Q}(Z)$.

PROOF OF ALGORITHM 3. Line 3 semi-rectifies the algebraic system S defining the steady points of the system Σ . Line 5 performs the change of coordinates in Σ . This change of coordinates is legitimate for the following reason. As stated in the input, the elements of Θ are parameters, i.e.

Algorithm 3 SemiRectifySteadyPoints(Σ, Θ, P)

Input: A system of ODEs Σ of the form $\dot{Z} = F(Z)$ with $p \geq e$ parameters.

A list of parameters assumed positive $\Theta = (z_1, \dots, z_e)$.

A list of remaining coordinates $P = (z_{e+1}, \dots, z_n)$.

Output: A semi-rectified system of ODEs $\tilde{\Sigma}$ of the form $\tilde{\dot{Z}} = \tilde{F}(\tilde{Z})$ obtained by rewriting Σ in the new coordinates $\tilde{Z}$.

An algebraic system $\tilde{S}$ defining the steady points of $\tilde{\Sigma}$ which is free of parameters of V .

A monomial map $\tilde{Z} = \Phi(Z)$ satisfying **(H)**.

A sublist V of $\tilde{Z}$ such that $\tilde{S}$ is semi-rectified for each element of V .

- 1: #Semi-rectification of the steady points
- 2: $S \leftarrow$ the numerators of $F(Z)$;
- 3: $[\tilde{S}, \tilde{Z} = \Phi(Z), V]$
 $\quad \quad \quad := \text{SemiRectifyAlgebraicSystem}(S, \Theta, P);$
- 4: #Computation of semi-rectified system of ODEs
- 5: $\tilde{\Sigma} \leftarrow \overline{\Phi^{-1}(\tilde{Z})} = F(\Phi^{-1}(\tilde{Z}));$
- 6: rewrite $\tilde{\Sigma}$ in the form $\tilde{\dot{Z}} = \tilde{F}(\tilde{Z});$
- 7: **return** $[\tilde{\Sigma}, \tilde{S}, \tilde{Z} = \Phi(Z), V];$

$\dot{z}_i = 0$ for $1 \leq i \leq e$. Since SemiRectifyAlgebraicSystem is called with Θ at line 3, the state variables (i.e the z_i for $p+1 \leq i \leq n$) are transformed using $\tilde{z}_i = z_i \prod_{k=1}^e z_k^{C_{k,i}}$ due to the third point of the condition **(H)**. Therefore, for each state variable, one has $\dot{\tilde{z}}_i = \dot{z}_i \prod_{k=1}^e z_k^{C_{k,i}}$. This explains that the system $\tilde{\Sigma}$ built at line 5 is a differential system with some non-zero extra multiplicative terms in front of some $\tilde{z}_i$. Those non-zero multiplicative terms are removed at line 6. Moreover, since those extra multiplicative terms are zero, $\tilde{S}$ defines the steady points of $\tilde{\Sigma}$ and is free of V . Roughly speaking, this means that taking the steady points and applying the monomial map to the differential system Σ are two operations that commute. $\square$

After calling the algorithm 3, the element of V are made free in the variety of the steady points of $\tilde{\Sigma}$. However, they are (a priori) still involved in the differential system $\tilde{\Sigma}$. Following the spirit of lemma 2.5, it appears that the elements of V are, in some sort, put in factor in the right hand side of Σ .

EXAMPLE 4.1. Let us illustrate the semi-rectification procedure of steady points of ODEs systems on the following academic example (with $a > 0$ and $b > 0$):

$$\begin{cases} \dot{x} = by^2 - a, \\ \dot{y} = ax - b \end{cases} \quad (29)$$

defined on the coordinate set $Z = (a, b, x, y)$. Remark that the two differential operators given in (12) are not scalings of the whole differential system meaning that they cannot be used, for example, for its reduction (see [4]). On the other hand, these two differential operators correspond to scalings of the algebraic system that defines its steady points i.e. of (8). Thus they can be used for the semi-rectification of this algebraic system. Following exactly the example 3.14, one can find the expressions of the new coordinates given

in (26). By rewriting the system (29) in these new coordinates, one obtains:

$$\begin{cases} \dot{\tilde{x}} = (\tilde{y}^2 - 1) \frac{\tilde{a}^4}{\tilde{b}^2}, \\ \dot{\tilde{y}} = (\tilde{x} - 1) \frac{\tilde{b}^3}{\tilde{a}}. \end{cases} \quad (30)$$

Observe that this new differential system depends on the parameters $\tilde{a}$ and $\tilde{b}$ but the algebraic system that defines its steady points is $\tilde{a}$ -semi-rectified and $\tilde{b}$ -semi-rectified.

The algorithm 3 can be applied to medium size (about twenty coordinates) differential systems. It decreases the number of parameters of their steady points expressions if the system possesses appropriate scalings. The following section illustrates this semi-rectification on a medium size system of ODEs coming from an example in biology.

5. EXAMPLE

Let us consider a model that represents a genetic network involving a single gene regulated by a pentamer of its own protein (see equation (1.5) in [17] with $n = 5$):

$$\begin{cases} \dot{G} = \gamma_0 - G - K_4 G P^5, \\ \dot{M} = (\gamma_0 - G) \rho_b + \rho_f G - \delta_M M, \\ \dot{P} = \frac{5(\gamma_0 - G) - 5 K_4 G P^5 - \delta_P P + \beta M}{1 + \sum_{i=1}^4 (i+1)^2 K_i P^i}. \end{cases} \quad (31)$$

The variable G represents the gene. This gene is transcribed into an mRNA denoted by M which is translated into a protein P . This regulatory protein forms a pentamer. The network also includes the degradation of the mRNA and the protein. Greek letters and K_i for all i in $\{1, \dots, 4\}$ represent parameters that are assumed positive. The steady points expressions of (31) depend on 7 parameters. The semi-rectification of these expressions leads to the following system of ODEs:

$$\begin{cases} \dot{\tilde{G}} = \tilde{\gamma}_0 - \tilde{G} - \tilde{K}_4 \tilde{G} \tilde{P}^5, \\ \dot{\tilde{M}} = ((\tilde{\gamma}_0 - \tilde{G}) \tilde{\rho}_b + \tilde{\rho}_f \tilde{G} - \tilde{M}) \tilde{\delta}_M, \\ \dot{\tilde{P}} = \frac{(5(\tilde{\gamma}_0 - \tilde{G}) - 5 \tilde{K}_4 \tilde{G} \tilde{P}^5 - \tilde{P} + \tilde{M}) \tilde{\delta}_P}{1 + \sum_{i=1}^4 (i+1)^2 \tilde{K}_i \tilde{P}^i}. \end{cases} \quad (32)$$

The relationships between the coordinates of (32) and these of (31) can be expressed by the following change of coordinates:

$$\tilde{\rho}_b = \frac{\rho_b \beta}{\delta_M}, \tilde{\rho}_f = \frac{\rho_f \beta}{\delta_M}, \tilde{\gamma}_0 = \frac{\gamma_0}{\delta_P}, \tilde{G} = \frac{G}{\delta_P}, \tilde{M} = \frac{M \beta}{\delta_P} \quad (33)$$

and all other coordinates remain the same.

This procedure considerably simplifies the steady points expressions that now depend on 4 parameters in (32). Remark that the parameter K_4 remains in the resulting systems even if it is given at the beginning of the positive parameters list. This is because the algebraic system defining the steady points of (31) does not possess any scalings acting on K_4 . After the semi-rectification the steady points do not depend on the free parameters $(\beta, \tilde{\delta}_M, \tilde{\delta}_P)$ anymore. We earned 3 freedom degrees for later computations.

Here are the associated MABSys commands assuming that the variable `Model` contains the description of the original system (31).

```

> Theta := [K4,K3,K2,K1,beta,deltaM,deltaP,rhof,rhof,gamma0];
  Theta := [K4, K3, K2, K1, beta, deltaM, deltaP, rhof, rhof, gamma0]

> RemainingCoords := [G,M,P];
  RemainingCoords := [G, M, P]

> out := SemiRectifySteadyPoints(Model, Theta, RemainingCoords):

> out[1,1];
  d
  [- G(t) = gamma0 - G(t) - K4 G(t) P(t) ,
  dt
  - M(t) = (rhof gamma0 - rhof G(t) + rhof G(t) - M(t)) deltaM,
  dt
  d
  deltaP (5 gamma0 - 5 G(t) - 5 K4 G(t) P(t) - P(t) + M(t))
  dt -----]
  1 + 4 K1 P(t) + 9 K2 P(t) + 16 K3 P(t) + 25 K4 P(t)

> out[1,2];
  5
  [gamma0 - G - K4 G P , rhof gamma0 - rhof G + rhof G - M,
  5
  5 gamma0 - 5 G - 5 K4 G P - P + M
  -----]
  1 + 4 K1 P + 9 K2 P + 16 K3 P + 25 K4 P

> out[1,3];
  rhof beta      rhof beta      gamma0      G      M beta
  [rhof-----, rhof-----, gamma0-----, G-----, M-----]
  deltaM      deltaM      deltaP      deltaP      deltaP

> out[1,4];
  [beta, deltaM, deltaP]

```

Observe that the original and the simplified systems coordinates are denoted by the same notation for the sake of computational clarity. The output must be interpreted as in (32) and (33).

REMARK 5.1. *If one uses the triangularization option, the steady points of the resulting system depend on only 2 parameters. Even if in theory the complexity of our algorithms increases with this option, it can be very useful in practice. For the computations, we used a computer with AMD Athlon(tm) Dual Core Processor 5400B, 3.4 GiB memory and Ubuntu 8.04 Hardy Heron as operating system. Our computations are instantaneous. It took 0.8 second without and 1 second with this triangularization option.*

6. CONCLUSION

We have presented two algorithms for the semi-rectification of algebraic systems and simplification of steady points expressions of a differential system. To our knowledge, it is the first time that one uses the scalings of an algebraic system that defines the steady points of a differential system for its symbolic simplification process. It is also important to underline that our method, despite the restrictions, can be efficiently (polynomial time complexity) used without any necessary acquaintance about Lie symmetries. In the future, we hope to improve our method by considering more general change of coordinates and symmetries.

7. REFERENCES

[1] F. Boulier, M. Lefranc, F. Lemaire, P.-E. Morant, and A. Ürgüplü. On proving the absence of oscillations in models of genetic circuits. In K. H. H. Anai and T. Kutsia, editors, *Proceedings of Algebraic Biology 2007*, volume 4545 of *LNCS*, pages 66–80. Springer Verlag Berlin Heidelberg, 2007.

[2] F. Boulier, F. Lemaire, A. Sedoglavic, and A. Ürgüplü. Towards an Automated Reduction Method for Polynomial ODE Models in Cellular Biology. *Mathematics in Computer Science, Special issue Symbolic Computation in Biology*, 2(3):443–464, March 2009.

[3] E. Cartan. *La méthode du repère mobile, la théorie des groupes continus et les espaces généralisés*. Exposés de géométrie – 5. Hermann, Paris, 1935.

[4] M. Fels and P. J. Olver. Moving coframes. II. Regularization and theoretical foundations. *Acta Applicandae Mathematicae*, 55(2):127–208, January 1999.

[5] E. Hubert. AIDA Maple package: Algebraic Invariants and their Differential Algebras, 2007. www.sop.inria.fr/members/Evelyn.Hubert/aida/.

[6] É. Hubert and A. Sedoglavic. Polynomial Time Nondimensionalisation of Ordinary Differential Equations via their Lie Point Symmetries. <http://hal.inria.fr/inria-00001251/en/>, 2006.

[7] R. Khanin. Dimensional Analysis in Computer Algebra. In B. Mourrain, editor, *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, pages 201–208, London, Ontario, Canada, July 22–25 2001. ACM, ACM press.

[8] F. Lemaire, M. Moreno Maza, and Y. Xie. The RegularChains library in MAPLE 10. In I. S. Kotsireas, editor, *The MAPLE conference*, pages 355–368, 2005.

[9] F. Lemaire and A. Ürgüplü. Modeling and Analysis of Biological Systems, 2008. Maple package (available at www.lifl.fr/~urguplu).

[10] E. Mansfield. Indiff: a Maple package for over determined differential systems with Lie symmetry, 2001.

[11] M. S. Okino and M. L. Mavrouniotis. Simplification of Mathematical Models of Chemical Reaction Systems. *Chemical Reviews*, 98(2):391–408, March/April 1998.

[12] P. J. Olver. *Applications of Lie groups to differential equations*, volume 107 of *Graduate Texts in Mathematics*. Springer Verlag, second edition, 1993.

[13] P. J. Olver. *Equivalence, Invariants, and Symmetry*. Cambridge University Press, 1995.

[14] A. Sedoglavic. Reduction of Algebraic Parametric Systems by Rectification of their Affine Expanded Lie Symmetries. *Proceedings of Algebraic Biology 2007 – Second International Conference*, 4545:277–291, July 2007. <http://hal.inria.fr/inria-00120991>.

[15] A. Sedoglavic and A. Ürgüplü. Expanded Lie Point Symmetry, 2007. Maple package (available at www.lifl.fr/~urguplu).

[16] H. Stephani. *Differential equations*. Cambridge University Press, 1st edition, 1989.

[17] A. Ürgüplü. *Contribution to Symbolic Effective Qualitative Analysis of Dynamical Systems; Application to Biochemical Reaction Networks*. PhD thesis, University of Lille 1, January, 13th 2010.

Absolute Factoring of Non-holonomic Ideals in the Plane

D. Grigoriev

CNRS, Mathématiques, Université de Lille, 59655, Villeneuve d'Ascq, France,
e-mail: dmitry.grigoryev@math.univ-lille1.fr,
website: <http://logic.pdmi.ras.ru/~grigorev>

F. Schwarz

FhG, Institut SCAI, 53754 Sankt Augustin, Germany,
e-mail: fritz.schwarz@scai.fraunhofer.de
website: www.scai.fraunhofer.de/schwarz.0.html

ABSTRACT

We study *non-holonomic* overideals of a left differential ideal $J \subset F[\partial_x, \partial_y]$ in two variables where F is a differentially closed field of characteristic zero. One can treat the problem of finding non-holonomic overideals as a generalization of the problem of factoring a linear partial differential operator. The main result states that a principal ideal $J = \langle P \rangle$ generated by an operator P with a separable symbol $\text{symp}(P)$ has a finite number of maximal non-holonomic overideals; the symbol is an algebraic polynomial in two variables. This statement is extended to non-holonomic ideals J with a separable symbol. As an application we show that in case of a second-order operator P the ideal $\langle P \rangle$ has an infinite number of maximal non-holonomic overideals iff P is essentially ordinary. In case of a third-order operator P we give sufficient conditions on $\langle P \rangle$ in order to have a finite number of maximal non-holonomic overideals. In the Appendix we study the problem of finding non-holonomic overideals of a principal ideal generated by a second order operator, the latter being equivalent to the Laplace problem. The possible application of some of these results for concrete factorization problems is pointed out.

Categories and Subject Descriptors

G.4 [Mathematical software]: Computer applications.

General Terms

Algorithms

Keywords

Differential non-holonomic overideals, Newton polygon, formal series solutions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

1. FINITENESS OF THE NUMBER OF MAXIMAL NON-HOLONOMIC OVER-IDEALS OF AN IDEAL WITH SEPARABLE SYMBOL

Let F be a differentially closed field (or universal differential field in terms of [8], [9]) with derivatives ∂_x and ∂_y ; let $P = \sum_{i,j} p_{i,j} \partial_x^i \partial_y^j \in F[\partial_x, \partial_y]$ be a partial differential operator of order n . Considering e.g. the field of rational functions $\mathbb{Q}(x, y)$ as F is a quite different issue. The *symbol* is defined by $\text{symp}(P) = \sum_{i+j=n} p_{i,j} v^i w^j$; it is a homogeneous algebraic polynomial of degree n in two variables. The degree of its Hilbert-Kolchin polynomial $ez + e_0$ is called its *differential type*; its leading coefficient is called the *typical differential dimension* [8]. A left ideal $I \subset F[\partial_x, \partial_y]$ is called *non-holonomic* if its differential type equals 1. We study maximal non-holonomic overideals of a principal ideal $\langle P \rangle \subset F[\partial_x, \partial_y]$. Obviously there is an infinite number of maximal *holonomic* overideals of $\langle P \rangle$: for any solution $u \in F$ of $Pu = 0$ we get a holonomic overideal $\langle \partial_x - u_x/u, \partial_y - u_y/u \rangle \supset \langle P \rangle$. We assume w.l.o.g. that $\text{symp}(P)$ is not divisible by ∂_y ; otherwise one can make a suitable transformation of the type $\partial_x \rightarrow \partial_x, \partial_y \rightarrow \partial_y + b\partial_x, b \in F$. In fact choosing b from the subfield of constants of F is possible.

Clearly, factoring an operator P can be viewed as finding principal overideals of $\langle P \rangle$; we refer to factoring over a universal field F as *absolute factoring*. Overideals of an ideal in connection with Loewy and primary decompositions were considered in [6].

Following [4] consider a homogeneous polynomial ideal $\text{symp}(I) \subset F[v, w]$ and attach a homogeneous polynomial $g = \text{GCD}(\text{symp}(I))$ to I . Lemma 4.1 [4] states that $\deg(g) = e$. As above one can assume w.l.o.g. that w does not divide g .

We recall that the Ore ring $R = (F[\partial_y])^{-1} F[\partial_x, \partial_y]$ (see [1]) consists of fractions of the form $\beta^{-1}r$ where $\beta \in F[\partial_y]$, $r \in F[\partial_x, \partial_y]$, see [3], [4]. We also recall that one can represent $R = F[\partial_x, \partial_y] (F[\partial_y])^{-1}$, and two fractions are equal, $\beta^{-1}r = r_1\beta_1^{-1}$, iff $\beta r_1 = r\beta_1$ [3], [4].

For a non-holonomic ideal I denote ideal $\bar{I} = RI \subset R$. Since the ring R is left-euclidean (as well as right-euclidean) with respect to ∂_x over the skew-field $(F[\partial_y])^{-1} F[\partial_y]$, we conclude that the ideal $\bar{I}$ is principal. Let $\bar{I} = \langle r \rangle$ for suitable $r \in F[\partial_x, \partial_y] \subset R$ (cf. [4]). Lemma 4.3 [4] implies that $\text{symp}(r) = w^m g$ for a certain integer $m \geq 0$ where g is not divisible by w .

Now we expose a construction introduced in [4]. For a family of elements $f_1, \dots, f_k \in F$ and rational numbers $s_i \in \mathbb{Q}$, $1 > s_2 > \dots > s_k > 0$ we consider a D -module being a vector space over F with a basis $\{G^{(s)}\}_{s \in \mathbb{Q}}$ where the derivatives of

$$G^{(s)} = G^{(s)}(f_1, \dots, f_k; s_2, \dots, s_k)$$

are defined as

$$d_{x_i} G^{(s)} = (d_{x_i} f_1) G^{(s+1)} + (d_{x_i} f_2) G^{(s+s_2)} + \dots + (d_{x_i} f_k) G^{(s+s_k)}$$

for $i = 1, 2$ using the notations $d_{x_1} = \partial_x, d_{x_2} = \partial_y$.

Next we introduce series of the form

$$\sum_{0 \leq i < \infty} h_i G^{(s - \frac{i}{q})} \quad (1)$$

where q is the least common multiple of the denominators of $s_2, \dots, s_k$; one can view (1) as an analogue of Newton-Puiseux series for non-holonomic D -modules. Theorem 2.5 [4] states that for any linear divisor $v + aw$ of $\text{symp}(P)$ and any $f_1 \in F$ such that $(\partial_x + a\partial_y)f_1 = 0$ there exists a solution of $P = 0$ of the form (1); conversely, if (1) is a solution of $P = 0$ then $(\partial_x + a\partial_y)f_1 = 0$ for an appropriate divisor $v + aw$ of $\text{symp}(P)$. Furthermore, Proposition 4.4 [4] implies that any solution of the form (1) of $r = 0$ such that $(\partial_x + a\partial_y)f_1 = 0$ for suitable $a \in F$ (or equivalently $\partial_y f_1 \neq 0$) is also a solution of the ideal I ; then the appropriate linear form $v + aw$ is a divisor of g , and the inverse holds as well.

In [5] we have designed an algorithm for factoring an operator P in case of $\text{symp}(P)$ is separable. In particular, in this case there is only a finite number (less than 2^n) of different factorizations of P . Now we show a more general statement for overideals of $\langle P \rangle$.

THEOREM 1.1. *Let $\text{symp}(P)$ be separable. Then there exists at most $n = \text{ord}(P)$ maximal non-holonomic overideals of $\langle P \rangle \subset F[\partial_x, \partial_y]$. Moreover, if there exists a non-holonomic overideal $I \supset \langle P \rangle$ with the attached polynomial $g = \text{GCD}(\text{symp}(I))$ then there exists a unique non-holonomic overideal, maximal among the ones with the attached polynomial equal g .*

PROOF. Let I be a non-holonomic ideal such that $I \supset \langle P \rangle$. Then $\beta P = r_1 r$ for suitable $\beta \in F[\partial_y]$, $r_1 \in F[\partial_x, \partial_y]$ and a polynomial $g = \text{GCD}(\text{symp}(I))$ attached to I is a divisor of $\text{symp}(P)$. We claim that for every pair of non-holonomic ideals $I_1, I_2 \supset \langle P \rangle$ to which a fixed polynomial g is attached, to their sum $I_1 + I_2$ also g is attached. Indeed, any solution of the form (1) of $P = 0$ such that $(v + aw)g$, is a solution of $r = 0$ as well due to Lemma 4.2 [4] (cf. Proposition 4.4 [4]) taking into account that $\text{symp}(P)$ is separable, hence it is also a solution of I as it was shown above and by the same token is a solution of both I_1 and I_2 (in particular $I_1 + I_2$ is also non-holonomic). The claim is established.

Thus among non-holonomic overideals $I \supset \langle P \rangle$ to which a given polynomial $g | \text{symp}(P)$ is attached, there is a unique maximal one. Now take two maximal non-holonomic overideals $I, I' \supset \langle P \rangle$ to which polynomials g, g' are attached, respectively. Then g, g' are reciprocally prime. Indeed, if $v + aw$ divides both g, g' then arguing as above one can verify that (1) is a solution of $I + I'$, i.e. the latter ideal is non-holonomic which contradicts to maximality of I, I' . Theorem is proved. $\square$

COROLLARY 1.2. *Let $\text{symp}(P)$ be separable. Suppose that there exist maximal non-holonomic overideals $I_1, \dots, I_l \supset \langle P \rangle$ such that for the respective attached polynomials $g_1, \dots, g_l$ the sum of their degrees $\deg(g_1) + \dots + \deg(g_l) \geq n$. The $\langle P \rangle = I_1 \cap \dots \cap I_l$.*

PROOF. As it was shown in the proof of Theorem 1.1, polynomials $g_j | \text{symp}(P)$, $1 \leq j \leq l$ are pairwise reciprocally prime, hence $g_1 \dots g_l = \text{symp}(P)$. Moreover it was established in the proof of Theorem 1.1 that every solution of $P = 0$ of the form (1) such that $(\partial_x + a\partial_y)f_1 = 0$, is a solution of a unique I_j for which $(u + aw) | g_j$; thus every solution of $P = 0$ of the form (1) is also a solution of $I_1 \cap \dots \cap I_l$. Therefore the typical differential dimension of ideal the $I_1 \cap \dots \cap I_l$ equals n (cf. Lemma 4.1 [4]). On the other hand, any overideal of a principal ideal $\langle P \rangle$ of the same typical differential dimension coincides with $\langle P \rangle$; one can verify it by comparing their Janet bases [10]. (We briefly recall that operators $P_1, \dots, P_s \in F[\partial_x, \partial_y]$ form a Janet basis of the ideal $\langle P_1, \dots, P_s \rangle$ if for any element $P \in \langle P_1, \dots, P_s \rangle$ its highest derivative $ld(P)$ is divided by one of $ld(P_i)$, $1 \leq i \leq s$.) $\square$

REMARK 1.3. *One can extend Theorem 1.1 to non-holonomic ideals J such that the homogeneous polynomial $\text{GCD}(\text{symp}(J))$ is separable: namely, there exists a finite number of maximal non-holonomic overideals $I \supset J$.*

2. NON-HOLONOMIC OVERIDEALS OF A SECOND-ORDER LINEAR PARTIAL DIFFERENTIAL OPERATOR

In this Section we study the structure of overideals of $\langle P \rangle$ when $n = \text{ord}(P) = 2$. The case of separable $\text{symp}(P)$ is covered by Theorem 1.1.

PROPOSITION 2.1. *Any principal ideal $\langle P \rangle$ for a second-order operator $P = \partial_y^2 + p_1 \partial_x + p_2 \partial_y + p_3$ with non-separable $\text{symp}(P)$ has*

- i) no proper non-holonomic overideals in case $p_1 \neq 0$;
- ii) an infinite number of maximal non-holonomic overideals in case $p_1 = 0$.

PROOF. Let $\text{symp}(P)$ be non-separable. Then applying a transformation of the type $\partial_x \rightarrow b_1 \partial_x + b_2 \partial_y$, $\partial_y \rightarrow b_3 \partial_x + b_4 \partial_y$ for suitable $b_1, b_2, b_3, b_4 \in F$ one can assume w.l.o.g. that $P = \partial_y^2 + p_1 \partial_x + p_2 \partial_y + p_3$; it would be interesting to find out when one can carry out these transformations algorithmically. First let $p_1 = 0$. Then P is essentially ordinary, i.e. becomes ordinary after a transformation as above, and for any solution $u \in F$ of the equation $P = 0$ we get a non-holonomic overideal $\langle \partial_y - u_y/u \rangle \supset \langle P \rangle$. Now suppose that $p_1 \neq 0$. Then P is irreducible (see e. g. Corollary 7.1 [4]). Moreover we claim that $\langle P \rangle$ has at most one maximal non-holonomic overideal. Let $I \supset \langle P \rangle$ be a non-holonomic overideal. Choosing arbitrary non-zero elements $b_1, b_2 \in F$ denote the derivation $d = b_1 \partial_x + b_2 \partial_y$. Similar to the proof of Theorem 1.1 there exists $r \in F[d, \partial_y] = F[\partial_x, \partial_y]$ such that $\langle r \rangle = IR_1 \subset R_1 = (F[d])^{-1} F[d, \partial_y]$. Then $\beta P = r_1 r$ for suitable $\beta \in F[d]$, $r_1 \in F[d, \partial_y]$ and $\text{symp}(r) = (b_1 v + b_2 w)^m g$ for an integer m and $g | w^2$. If $g = 1$ then I cannot be non-holonomic because of Proposition 4.4 [4] (cf. above). If $g = w^2$ then similar to the proof of Corollary 1.2 one can show that the only non-holonomic overideal of $\langle P \rangle$ among ones to which polynomial w^2 is attached, is just $\langle P \rangle$ itself. It remains to consider the case

$g = w$. Applying the Newton polygon construction from [4] to equation $r = 0$ and a divisor w of $\text{symb}(r)$, one obtains a solution of the form (1) of $r = 0$ with $G = G(x)$, thereby it is a solution of $P = 0$. On the other hand, applying the Newton polygon construction from [4] to equation $P = 0$, one gets at its first step $f_1 = x$ and at the second step f_2 which fulfils equation $(\partial_y f_2)^2 + p_1 = 0$ and f_2 corresponds to the edge of the Newton polygon with endpoints $(0, 2)$ and $(1, 0)$, so with the slope $1/2$. This provides a solution of equation $P = 0$ of the form (1) with $G = G(x, f_2; 1/2)$, therefore the equation $P = 0$ has no solutions of the form (1) with $G = G(x)$. The achieved contradiction shows that there are no non-holonomic overideals I with attached polynomial w , this completes the proof of the claim. $\square$

3. ON NON-HOLONOMIC OVERIDEALS OF A THIRD-ORDER OPERATOR

Now we study overideals of $\langle P \rangle$ where the order $n = \text{ord}(P) = 3$. Due to Theorem 1.1 it remains to consider non-separable $\text{symb}(P)$. In [4] an algorithm has been designed for factoring P ; a few explicit calculations for factoring P are provided in [7].

PROPOSITION 3.1. *Let P be a third-order operator with a non-separable $\text{symb}(P)$.*

i) When $\text{symb}(P)$ has two different linear divisors, one of which of multiplicity 2, then we can assume w.l.o.g. that

$$P = \partial_x \partial_y^2 + p_0 \partial_x^2 + p_1 \partial_x \partial_y + p_2 \partial_y^2 + p_3 \partial_x + p_4 \partial_y + p_5.$$

If $p_0 \neq 0$ then $\langle P \rangle$ has at most two maximal non-holonomic overideals. Moreover if there exist two different maximal non-holonomic overideals $I_1, I_2 \supset \langle P \rangle$ then $\langle P \rangle = I_1 \cap I_2$;

ii) When $\text{symb}(P)$ has a single linear divisor of multiplicity 3 we can assume w.l.o.g. that

$$P = \partial_y^3 + p_0 \partial_x^2 + p_1 \partial_x \partial_y + p_2 \partial_y^2 + p_3 \partial_x + p_4 \partial_y + p_5.$$

If either $p_0 \neq 0$, either $p_2 \neq 0$ or $p_3 \neq 0$ then $\langle P \rangle$ has at most two maximal non-holonomic overideals. Moreover if there exist two different maximal non-holonomic overideals $I_1, I_2 \supset \langle P \rangle$ then $\langle P \rangle = I_1 \cap I_2$. Otherwise $\langle P = \partial_y^3 + p_2 \partial_y^2 + p_4 \partial_y + p_5 \rangle$ has an infinite number of maximal non-holonomic overideals.

PROOF. Case *i*) First let $\text{symb}(P)$ have two linear divisors; therefore one can assume w.l.o.g. (see above) that w is its divisor of multiplicity 2 and v is its divisor of multiplicity 1. One can write

$$P = \partial_x \partial_y^2 + p_0 \partial_x^2 + p_1 \partial_x \partial_y + p_2 \partial_y^2 + p_3 \partial_x + p_4 \partial_y + p_5.$$

Suppose that $p_0 \neq 0$. The Newton polygon construction from [4] applied to equation $P = 0$ and to divisor w of $\text{symb}(P)$, yields a solution of the form (1) of $P = 0$ with $f_1 = x$ at its first step. At its second step the construction yields f_2 which fulfils equation $(\partial_y f_2)^2 + p_0 = 0$ and which corresponds to the edge of the Newton polygon with endpoints $(1, 2)$, $(2, 0)$, so with the slope $1/2$. This provides $G = G(x, f_2; 1/2)$ in (1).

Let a non-holonomic ideal $I \supset \langle P \rangle$. Choose $d = b_1 \partial_x + b_2 \partial_y$ for non-zero $b_1, b_2 \in F$. As in the previous Section there exists $r \in F[d, \partial_y]$ such that $\langle r \rangle = R_1 I \subset R_5 = (F[d])^{-1} F[d, \partial_y]$. Then $\beta P = r_1 r$ for suitable $\beta \in F[d]$, $r_1 \in F[d, \partial_y]$. Rewrite $\text{symb}(r) = (b_1 v + b_2 w)^m g$ where $g|(vw^2)$. If either $g = w^2$ or $g = v$, one can argue as in the proof of

Theorem 1.1 and deduce that there can exist at most one maximal non-holonomic overideal of $\langle P \rangle$ with the property that the polynomial attached to the overideal is either w^2 or v . Similar to the proof of Corollary 1.2 one can verify that if there exist maximal non-holonomic overideals $I_2, I_1 \supset \langle P \rangle$ with attached polynomials w^2 and v , then $\langle P \rangle = I_1 \cap I_2$. As in Theorem 1.1 the existence of a maximal overideal with the attached polynomial w^2 or v follows from the existence of any non-holonomic overideal with the attached polynomial w^2 or v .

If either $g = w$ or $g = vw$ then applying the Newton polygon construction from [4] to equation $r = 0$ and divisor w of $\text{symb}(r)$, one obtains a solution of $r = 0$ (and thereby, of $P = 0$ due to Lemma 4.2 [4]) of the form (1) with $G = G(x)$ which contradicts to the supposition $p_0 \neq 0$ (see above). Thus, in case $p_0 \neq 0$ the ideal $\langle P \rangle$ has a finite number, less or equal than 2, of maximal non-holonomic overideals (similar to Theorem 1.1).

When $p_0 = 0$ this is not always true, say for $P = (\partial_x + b)(\partial_y^2 + b_3 \partial_y + b_4)$ (cf. case $n = 2$ in the previous Section). It would be interesting to clarify for which P this is still true.

Case *ii*) Now we consider the last case when $\text{symb}(P)$ has a unique linear divisor with multiplicity 3. As above one can assume w.l.o.g. that $\text{symb}(P) = w^3$, so

$$P = \partial_y^3 + p_0 \partial_x^2 + p_1 \partial_x \partial_y + p_2 \partial_y^2 + p_3 \partial_x + p_4 \partial_y + p_5.$$

Keeping the notations we get $\langle r \rangle = R_1 I$ and $\beta P = r_1 r$. Then $\text{symb}(r) = (b_1 v + b_2 w)^m g$ where $g|w^3$. If $g = w^3$ then arguing as in the proof of Corollary 1.2 we deduce that the only non-holonomic overideal of $\langle P \rangle$ to which polynomial w^3 is attached, is just $\langle P \rangle$ itself.

Let $g|w^2$. Applying the Newton polygon construction from [4] to equation $r = 0$ and linear divisor w of $\text{symb}(r)$ one gets a solution of $r = 0$ (and thereby of $P = 0$) with either $G = G(x)$ or $G = G(x, f_2; 1/2)$ where $\partial_y f_2 \neq 0$ (cf. above).

Application of the Newton polygon construction from [4] to equation $P = 0$ (and unique linear divisor w of $\text{symb}(P)$) at its first step provides $f_1 = x$. The second step requires a trial of cases. First let $p_0 \neq 0$. Then the second step yields f_2 which fulfils equation $(\partial_y f_2)^3 + p_0 = 0$ and which corresponds to the edge of the Newton polygon with endpoints $(0, 3)$, $(2, 0)$, so with the slope $2/3$. Thus we obtain a solution of the form (1) with $G = G(x, f_2, \dots; 2/3, \dots)$, hence $\langle P \rangle$ in case $p_0 \neq 0$ has no non-holonomic overideals with attached polynomial g being a divisor of w^2 (see above).

Now assume that $p_0 = 0$ and $p_1 \neq 0$. Then the second step provides solutions of $P = 0$ of the form (1) with two different possibilities. Either the Newton polygon construction chooses the vertical edge with endpoints $(1, 1)$, $(1, 0)$ as a leading edge at the second step, then it terminates at the second step yielding a solution of the form (1) with $G = G(x)$; we recall that in the construction from Section 2 [4] only edges with non-negative slopes are taken as leading ones and the construction terminates while taking a vertical edge, so with the slope 0, as a leading one, in particular the edge with endpoints $(1, 1)$, $(1, 0)$ is taken as a leading one regardless of whether the coefficient at point $(1, 0)$ vanishes. As the second possibility the construction yields a solution of the form (1) with $G = G(x, f_2, \dots; 1/2, \dots)$ where $f_2 \neq 0$ fulfils equation $(\partial_y f_2)^3 + p_1 \partial_y f_2 = 0$ corresponding to the edge of the Newton polygon with endpoints $(0, 3)$, $(1, 1)$, so with the slope $1/2$. One can suppose w.l.o.g. that the Newton polygon construction terminates at its third step

(thereby $G = G(x, f_2; 1/2)$), otherwise $\langle P \rangle$ cannot have a non-holonomic overideal to which a divisor g of w^2 is attached (see above).

If $g = w^2$ then any solution H_2 of $P = 0$ of the form (1) with $G = G(x, f_2; 1/2)$ is a solution of $r = 0$ because otherwise $rH_2 \neq 0$, being also of the form (1) with $G = G(x, f_2; 1/2)$, cannot be a solution of $r_1 = 0$ taking into account that $\text{symb}(r_1)$ does not divide on w^2 (cf. Lemma 4.2 [4]). Else if $g = w$ then $rH_2 \neq 0$ (again taking into account that $\text{symb}(r)$ does not divide on w^2) and therefore $r_1(rH_2) = 0$. Hence for a solution H_1 of $P = 0$ of the form (1) with $G = G(x)$ (see above) we have $rH_1 = 0$ since otherwise rH_1 being also of the form (1) with $G = G(x)$ cannot be a solution of $r_1 = 0$ (again cf. Lemma 4.2 [4]). Then arguing as in the proof of Theorem 1.1 one concludes that in case $p_0 = 0$ and $p_1 \neq 0$ ideal $\langle P \rangle$ can have at most two maximal non-holonomic overideals with attached polynomials w and w^2 . Similar to the proof of Corollary 1.2 (cf. the preceding Subsection) one can verify that if there exist maximal non-holonomic overideals $I_1, I_2 \supset \langle P \rangle$ with attached polynomials w and w^2 , then $\langle P \rangle = I_1 \cap I_2$. As in Theorem 1.1 the existence of a maximal overideal with the attached polynomial w (or respectively, w^2) follows from the existence of any non-holonomic overideal with the attached polynomial w or w^2 .

Furthermore, let $p_0 = p_1 = 0$, $p_3 \neq 0$. Then as in case $p_0 \neq 0$ we argue that the second step of the Newton polygon construction applied to equation $P = 0$ yields f_2 which fulfils equation $(\partial_y f_2)^3 + p_4 = 0$ and which corresponds to the leading edge of the Newton polygon with end-points $(0, 3)$, $(1, 0)$, so with the slope $1/3$. Thus the Newton polygon construction yields a solution of $P = 0$ of the form (1) with $G = G(x, f_2, \dots; 1/3, \dots)$ and again $\langle P \rangle$ in case $p_0 = p_1 = 0$, $p_3 \neq 0$ under consideration has no non-holonomic overideals with an attached polynomial being a divisor of w^2 .

Finally, when $p_0 = p_1 = p_3 = 0$ the ideal $\langle P = \partial_y^3 + p_1 \partial_y^2 + p_3 \partial_y + p_5 \rangle$ has an infinite number of maximal non-holonomic overideals; this is similar to the second-order case $P = \partial_y^2 + p_4 \partial_y + p_5$, see above. $\square$

A few examples applying the preceding result are given next.

EXAMPLE 1. The operator

$$L \equiv \partial_{yy} + x\partial_x + \partial_y + y$$

is immediately recognized as absolutely irreducible by case *i*) of PROPOSITION 2.1 because $p_1 \neq 0$.

EXAMPLE 2. Consider the operator

$$L \equiv \partial_{xyy} + \partial_{xx} + y\partial_{yy} + (y+1)\partial_x + 2\partial_y + y.$$

Due to $p_0 = 1$, case *i*) of the above proposition applies. In fact, there is only a single first-order right factor as may be seen from

$$L = (\partial_{yy} + \partial_x + 1)(\partial_x + y);$$

this decomposition may be obtained by using the function `FirstOrderRightFactors` provided on the website www.alltypes.de [11].

EXAMPLE 3. Case *ii*) of PROPOSITION 3.1 applies to the operator

$$L \equiv \partial_{yyy} + \frac{x}{y^2} \partial_{xy} + \left(1 + \frac{2}{y}\right) \partial_{yy} + \frac{x(y-2)}{y^3} \partial_x + \frac{2y-3}{y^2} \partial_y - \frac{y-2}{y^3};$$

although $p_0 = 0$, due to $p_2 \neq 0$ and $p_3 \neq 0$ the operator can have at most two different right factors. It turns out that there are no first-order right factors at all.

It is a challenge to design an algorithm which produces non-holonomic overideals of a given differential ideal $J \subset F[\partial_x, \partial_y]$ in general. If the goal is solving linear pde's attached to these operators, $F = \mathbb{Q}(x, y)$ is of particular interest. Some of the results reported in this article may be applied for obtaining a partial answer; e.g. by case *i*) of PROPOSITION 2.1 it may be possible to exclude the existence of any factor very efficiently.

Appendix. Explicit formulas for Laplace transformation

We exhibit a short exposition and explicit formulas for the Laplace transformation [2]. Let $Q = \partial_{xy} + a\partial_x + b\partial_y + c$ be a second-order operator which has its Laplace divisor $L_n = \sum_{0 \leq i \leq n} l_i \partial_x^i$ of order n , i. e. Q, L_n form a Janet basis of ideal $\langle Q, L_n \rangle$. Hence

$$PQ = (\partial_y + a)L_n \quad (2)$$

for a suitable $P = \sum_{0 \leq i \leq n-1} p_i \partial_x^i$. (This form of P is obtained by comparing the highest terms which divide on ∂_x^n in (2).)

If a Laplace divisor exists then $\langle Q, L_n \rangle$ is a proper non-holonomic overideal of $\langle Q \rangle$. Conversely, one can show (cf. [2]) that if $\langle Q \rangle$ has a proper non-holonomic overideal then there exists either a Laplace divisor L_n (for a suitable n) or a Laplace divisor of the form $\sum_{0 \leq i \leq n} t_i \partial_y^i$ with respect to ∂_y . That is why the problem of searching for a Laplace divisor is equivalent to finding non-holonomic proper overideals of $\langle Q \rangle$.

Open question: is there an algorithm which decides for a given Q whether it has a Laplace divisor? In particular, an upper bound on n would suffice for an algorithm.

Comparing the highest terms in (2) which divide on ∂_y , we get that $L_n = P(\partial_x + b)$. Thus

$$PQ = (\partial_y + a)P(\partial_x + b). \quad (3)$$

We have $Q \neq (\partial_y + a)(\partial_x + b)$ iff $0 \neq ab + b_y - c \equiv K_0$.

LEMMA 3.2. If $K_0 \neq 0$ then there are unique B, C such that

$$(\partial_x + B)Q = (d_{xy} + a\partial_x + B\partial_y + C)(\partial_x + b) \quad (4)$$

PROOF. (4) is equivalent to an algebraic linear system in B, C ,

$$aB - C = b_y + ab - a_x - c, \quad (5)$$

$$(c - b_y)B - bC = b_{xy} + ab_x - c_x \quad (6)$$

$\square$

Therefore (3) holds iff $P = P_1(\partial_x + B)$ by means of dividing P by $\partial_x + B$ with remainder. Substituting the latter equality to (3) and making use of (4) we obtain the equality

$$P_1(\partial_{xy} + a\partial_x + B\partial_y + C) = (\partial_y + a)P_1(\partial_x + B). \quad (7)$$

Now (7) is similar to (3) but with the order $\text{ord}(P_1) = \text{ord}(P) - 1 = n - 1$ and a new second-order operator $Q_1 = \partial_{xy} + a\partial_x + B\partial_y + C$. Continuing this way we get the Laplace transformation with $K_1 = aB + B_y - C$ etc.

More uniformly denote $b_0 \equiv b$, $c_0 \equiv c$, then $b_1 \equiv B$, $c_1 \equiv C$, b_2, c_2 etc. obtained from Lemma 3.2. Denote

$$K_i \equiv ab_i + (b_i)_y - c_i, \quad Q_i \equiv \partial_{xy} + a\partial_x + b_i\partial_y + c_i.$$

COROLLARY 3.3. *There exists L_n satisfying (2) iff for the minimal m such that $K_m = 0$ we have $m \leq n$. In this case*

$$L_n = P_{n-m}(\partial_x + b_{m-1}) \cdots (\partial_x + b_0) \quad (8)$$

where $P_{n-m} = \sum_{0 \leq i \leq n-m} p_i \partial_x^i$ is an arbitrary operator of the order $n - m$ which fulfils

$$P_{n-m}(\partial_y + a) = (\partial_y + a)P_{n-m}. \quad (9)$$

For any order $n - m \geq 0$ such an operator P_{n-m} exists. The pair Q, L_n constitutes a Janet basis of the ideal $\langle Q, L_n \rangle$. The ideal $\langle Q, L_m \rangle$ is the unique maximal non-holonomic overideal of $\langle Q \rangle$ which corresponds to a divisor y of $\text{symb}(Q) = xy$ (see Theorem 1.1).

PROOF. Applying Laplace transformations as above, if $m > n$ we don't get a solution of (2) after n steps since (3) with $PQ_n = (\partial_y + a)P(\partial_x + b_n)$ would not have a solution with P of the order 0. If $m \leq n$ then successively following Laplace transformations we arrive to (8) in which (9) is obtained from equality $PQ_m = (\partial_y + a)P(\partial_x + b_m)$ (see (3)) and taking into account that $K_m = 0$. $\square$

ACKNOWLEDGEMENT The first author is grateful to the Max-Planck Institut für Mathematik, Bonn for its hospitality while writing this paper.

4. REFERENCES

- [1] J. E. Björk, Rings of differential operators, North-Holland, 1979.
- [2] E. Goursat, Leçon sur l'intégration des équations aux dérivées partielles, vol. I, II, A. Hermann, 1898.
- [3] D. Grigoriev, Weak Bézout Inequality for D-Modules, *J. Complexity* **21** (2005), 532-542.
- [4] D. Grigoriev, Analogue of Newton-Puiseux series for non-holonomic D-modules and factoring, *Moscow Math. J.* **9** (2009), 775-800.
- [5] D. Grigoriev, F. Schwarz, Factoring and solving linear partial differential equations, *Computing* **73** (2004), 179-197.
- [6] D. Grigoriev, F. Schwarz, Loewy and primary decomposition of D-Modules, *Adv. Appl. Math.* **38** (2007), 526-541.
- [7] D. Grigoriev, F. Schwarz, Loewy decomposition of linear third-order PDE's in the plane, Proc. Intern. Symp. Symbolic, Algebr. Comput., ACM Press, 277-286.
- [8] E. Kolchin, Differential Algebra and Algebraic Groups, Academic Press, New York, 1973.
- [9] M. van der Put, M. Singer, Galois theory of linear differential equations, Grundlehren der Mathematischen Wissenschaften, **328**, Springer, 2003.
- [10] F. Schwarz, Janet bases for symmetry groups, Groebner bases and applications, in London Math. Society, Lecture Note Ser. **251**, 221-234, Cambridge University Press, Cambridge, 1998.
- [11] F. Schwarz, ALLTYPES in the Web, ACM Communications in Computer Algebra, Vol. 42, No. 3, page 185-187(2008).

Algorithms for Bernstein–Sato Polynomials and Multiplier Ideals

Christine Berkesch
Department of Mathematics
Purdue University
cberkesch@purdue.edu

Anton Leykin
School of Mathematics
Georgia Institute of Technology
leykin@math.gatech.edu

ABSTRACT

The Bernstein–Sato polynomial (or global b -function) is an important invariant in singularity theory, which can be computed using symbolic methods in the theory of D -modules. After providing a survey of known algorithms for computing the global b -function, we develop a new method to compute the local b -function for a single polynomial. We then develop algorithms that compute generalized Bernstein–Sato polynomials of Budur–Mustață–Saito and Shibuta for an arbitrary polynomial ideal. These lead to computations of log canonical thresholds, jumping coefficients, and multiplier ideals. Our algorithm for multiplier ideals simplifies that of Shibuta and shares a common subroutine with our local b -function algorithm. The algorithms we present have been implemented in the D -modules package of the computer algebra system Macaulay2.

Categories and Subject Descriptors

G.0 [General]: Miscellaneous

General Terms

Algorithms

Keywords

Bernstein–Sato polynomial, log-canonical threshold, jumping coefficients, multiplier ideals, D -modules, V -filtration

1. Introduction

The multiplier ideals of an algebraic variety carry essential information about its singularities and have proven themselves a powerful tool in algebraic geometry. However, they are notoriously difficult to compute; nice descriptions are known only for very special families of varieties, such as monomial ideals and hyperplane arrangements [10, 19, 35, 27]. To briefly recall the definition of this invariant, let $X = \mathbb{C}^n$ with coordinates $\mathbf{x} = x_1, \dots, x_n$. For an ideal

$\langle \mathbf{f} \rangle = \langle f_1, \dots, f_r \rangle \subseteq \mathbb{C}[\mathbf{x}]$ and a nonnegative rational number c , the *multiplier ideal of $\mathbf{f}$ with coefficient c* is

$$\mathcal{J}(\mathbf{f}^c) = \left\{ h \in \mathbb{C}[\mathbf{x}] \mid \frac{|h|^2}{(\sum |f_i|^2)^c} \text{ is locally integrable} \right\}.$$

It follows from this definition that $\mathcal{J}(\mathbf{f}^c) \supseteq \mathcal{J}(\mathbf{f}^d)$ for $c \leq d$ and $\mathcal{J}(\mathbf{f}^0) = \mathbb{C}[\mathbf{x}]$ is trivial. The (global) *jumping coefficients* of $\mathbf{f}$ are a discrete sequence of rational numbers $\xi_i = \xi_i(\mathbf{f})$ with $0 = \xi_0 < \xi_1 < \xi_2 < \dots$ satisfying the property that $\mathcal{J}(\mathbf{f}^c)$ is constant exactly for $c \in [\xi_i, \xi_{i+1})$. In particular, the *log canonical threshold* of $\mathbf{f}$ is ξ_1 , denoted by $\text{lct}(\mathbf{f})$. This is the least rational number c for which $\mathcal{J}(\mathbf{f}^c)$ is nontrivial. The multiplier ideal $\mathcal{J}(\mathbf{f}^c)$ measures the singularities of the variety of $\mathbf{f}$ in X ; smaller multiplier ideals (and lower log canonical threshold) correspond to worse singularities. For an equivalent algebro-geometric definition and an introduction to this invariant, we refer the reader to [13, 14].

In this paper we develop an algorithm for computing multiplier ideals and jumping coefficients by way of an even finer invariant, Bernstein–Sato polynomials, or b -functions. The results of Budur *et al.* [6] provide other applications for our Bernstein–Sato algorithms, including multiplier ideal membership tests, an algorithm to compute jumping coefficients, and a test to determine if a complete intersection has at most rational singularities.

The first b -function we consider, the *global Bernstein–Sato polynomial* of a hypersurface, was introduced independently by Bernstein [4] and Sato [29]. This univariate polynomial plays a central role in the theory of D -modules (or algebraic analysis), which was founded by, amongst others, Kashiwara [11] and Malgrange [17]. Moreover, the jumping coefficients of $\mathbf{f}$ that lie in the interval $(0, 1]$ are roots of its global Bernstein–Sato polynomial [7]; however, this b -function contains more information. Its roots need not be jumping coefficients, even if they are between 0 and 1 (see Example 6.1).

The Bernstein–Sato polynomial was recently generalized by Budur *et al.* [6] to arbitrary varieties. The maximal root of this *generalized Bernstein–Sato polynomial* provides a multiplier ideal membership test. Shibuta defined another generalization to compute explicit generating sets for multiplier ideals [32]. Our multiplier ideal algorithm employs the b -functions of Shibuta, which we call the *m-generalized Bernstein–Sato polynomial*. However, it circumvents primary decomposition and one elimination step through a syzygetic technique (see Algorithms 4.5 and 3.2). The correctness of our results relies heavily on the use of V -filtrations, as developed by Kashiwara and Malgrange [12, 18].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

D -module computations are made possible by Gröbner bases techniques in the Weyl algebra. The computation of the Bernstein–Sato polynomial was pioneered by Oaku in [24]. His algorithm was one of the first algorithms in algebraic analysis, many of which are outlined in the book by Saito *et al.* [28]. The computation of the *local Bernstein–Sato polynomial* was first addressed in the early work of Oaku [24], as well as the recent work of Nakayama [20], Nishiyama and Noro [21], and Schulze [30, 31]. Bahloul and Oaku [2] address the computation of local Bernstein–Sato ideals that generalize Bernstein–Sato polynomials. In this article we provide our version of the local algorithm for Bernstein–Sato polynomials, part of which is vital to our approach to computation of multiplier ideals.

There are several implementations of algorithms for global and local b -functions in kan/sm1 [33], Risa/Asir [23], and Singular [9]. One can find a comparison of performance in [15]. All of the algorithms in this article have been implemented and can be found in the D -modules package [16] of the computer algebra system Macaulay2 [8].

The first author was partially supported by NSF Grants DMS 0555319 and DMS 090112; the second author is partially supported by the NSF Grant DMS 0914802.

Outline

Section 2 surveys the known approaches for computing the global Bernstein–Sato polynomial, highlighting an algorithm of Noro [22]. In Section 3, we present an algorithm for computing the local Bernstein–Sato polynomial. Algorithms for the generalized Bernstein–Sato polynomial for an arbitrary variety, as introduced by Budur *et al.* [6], are discussed in Section 4, along with their applications. Based on the methods of Section 3, Section 5 considers the m -generalized Bernstein–Sato polynomial of Shibuta [32] and contains our algorithms for multiplier ideals.

2. Global Bernstein–Sato polynomials

Let K be a field of characteristic zero, and set $X = K^n$ and $Y = X \times K$ with coordinates $(\mathbf{x})$ and $(\mathbf{x}, t)$, respectively. We consider the n -th Weyl algebra $D_X = K\langle \mathbf{x}, \partial \rangle$ with generators $x_1, \dots, x_n$ and $\partial_{x_1}, \dots, \partial_{x_n}$, as well as $D_Y = K\langle \mathbf{x}, \partial_{\mathbf{x}}, t, \partial_t \rangle$, the Weyl algebra on Y . Define an action of D_Y on $N_f := K[\mathbf{x}][f^{-1}, s]f^s$ as follows: x_i and ∂_{x_i} act naturally for $i = 1, \dots, n$, and

$$\begin{aligned} t \cdot h(\mathbf{x}, s)f^s &= h(\mathbf{x}, s+1)f^s, \\ \partial_t \cdot h(\mathbf{x}, s)f^s &= -sh(\mathbf{x}, s-1)f^{-1}f^s, \end{aligned}$$

where $h \in K[\mathbf{x}][f^{-1}, s]$.

Let $\sigma = -\partial_t t$. For a polynomial $f \in K[\mathbf{x}]$, the *global Bernstein–Sato polynomial* of f , denoted b_f , is the monic polynomial $b(s) \in K[s]$ of minimal degree satisfying the equation

$$b(\sigma)f^s = Pf f^s \quad (2.1)$$

for some $P \in D_X\langle \sigma \rangle$.

There is an alternate definition for the global Bernstein–Sato polynomial in terms of V -filtrations. To provide this, we denote by $V^\bullet D_Y$ the V -filtration of D_Y along X , where $V^m D_Y$ is D_X -generated by the set $\{t^\mu \partial_t^\nu \mid \mu - \nu \geq m\}$. Let $i_f : X \rightarrow Y$ defined by $i_f(\mathbf{x}) = (\mathbf{x}, f(\mathbf{x}))$ be the graph of f .

The D -module direct image of $K[\mathbf{x}]$ along i_f is the module

$$M_f := (i_f)_+ K[\mathbf{x}] \cong K[\mathbf{x}] \otimes_K K\langle \partial_t \rangle$$

with actions of a vector field ξ on X and t ,

$$\begin{aligned} \xi(p \otimes \partial_t^\nu) &= \xi p \otimes \partial_t^\nu - (\xi f)p \otimes \partial_t^{\nu+1}, \\ t \cdot (p \otimes \partial_t^\nu) &= fp \otimes \partial_t^\nu - \nu p \otimes \partial_t^{\nu-1}, \end{aligned}$$

providing a D_Y -module structure. Notice that there is a canonical embedding of M_f into N_f , where s is identified with $-\partial_t t$.

With $\delta = 1 \otimes 1 \in M_f$, the global Bernstein–Sato polynomial b_f is equal to the minimal polynomial of the action of σ on the module $(V^0 D_Y)\delta / (V^1 D_Y)\delta$. We now survey three ways of computing this b -function.

2.1 By way of an annihilator

The global Bernstein–Sato polynomial $b_f(s)$ is the minimal polynomial of $\sigma := -\partial_t t$ modulo $\text{Ann}_{D_X[\sigma]} f^s + D_X[\sigma]f$, where $f^s \in N_f$. By the next result, this annihilator can be computed from the left D_Y -ideal

$$I_f = \left\langle t - f, \partial_1 + \frac{\partial f}{\partial x_1} \partial_t, \dots, \partial_n + \frac{\partial f}{\partial x_n} \partial_t \right\rangle.$$

THEOREM 2.1. [28, Theorem 5.3.4] *The ideal $\text{Ann}_{D[s]} f^s$ equals the image of $I_f \cap D[\sigma]$ under the substitution $\sigma \mapsto s$.*

2.2 By way of an initial ideal

This method makes use of $w = (0, 1) \in \mathbb{R}^n \times \mathbb{R}$, the elimination weight vector for X in Y .

THEOREM 2.2. *Let $b(x, s)$ be nonzero in the polynomial ring $K[x, s]$. Then $b(x, \sigma) \in (\text{in}_{(-w, w)} I_f) \cap K[x, \sigma]$ if and only if there exists $Q \in D[s]$ satisfying the functional equation $Qf^{s+1} = b(x, s)f^s$. In particular,*

$$\langle b_f(\sigma) \rangle = \text{in}_{(-w, w)} I_f \cap K[\sigma].$$

PROOF. The action of t on N_f is multiplication by f , hence, the existence of the functional equation is equivalent to $b(x, s) \in I_f + V^1 D_Y$. The result now follows from Theorem 2.1, which identifies s with σ . $\square$

The following algorithm provides a more economical way to compute the global b -function using linear algebra. By establishing a nontrivial K -linear dependency between normal forms $\text{NF}_G(s^i)$ with respect to a Gröbner basis G of $\text{in}_{(-w, w)} I_f$, where $0 \leq i \leq d$ and d is taken as small as possible, this algorithm bypasses elimination of $\partial_1, \dots, \partial_n$. This trick was used for the first time by Noro [22], where a modular method to speed up b -function computations is provided as well. We include the following algorithm for the convenience of the reader as a similar syzygetic approach will be used in Algorithms 3.2, 4.5, and 5.12. Note that the coefficients of the output are, in fact, rational, since the roots of a b -function are rational [11].

ALGORITHM 2.3. $b = \text{globalBFfunction}(f, P)$

Input: a polynomial $f \in K[\mathbf{x}]$.

Output: polynomial $b \in \mathbb{Q}[s]$ is the Bernstein–Sato polynomial of f .

$G \leftarrow$ Gröbner basis of $\text{in}_{(-w, w)} I_f$.

$d \leftarrow 0$.

repeat

$d \leftarrow d + 1$

until $\exists(c_0, \dots, c_d) \in \mathbb{Q}^{d+1}$ such that $c_d = 1$ and

$$\sum_{i=0}^d c_i \text{NF}_G(s^i) = 0.$$

return $\sum_{i=0}^d c_i s^i$.

This approach can be exploited in a more general setting to compute the intersection of a left ideal with a subring generated by one element as shown in [1].

2.3 By way of Briançon–Maisonobe

This approach, which is laid out in [5], computes the annihilator of f^s in an algebra of solvable type similar to, but different from, the Weyl algebra. This path has been explored by Castro-Jiménez and Ucha [36] and implemented in Singular [9] with a performance analysis given by Levandovskyy and Morales in [15] and recent improvements outlined in [1].

3. Local Bernstein–Sato polynomials

In this section, we provide an algorithm to compute the local Bernstein–Sato polynomial of f at a prime ideal of $K[\mathbf{x}]$, which is defined by replacing the use of D_X in (2.1) by its appropriate localization. Algorithms 3.1 and 3.2 use Theorem 2.2 to compute an ideal $E_b \subset K[\mathbf{x}]$ that describes the locus of points where the b -function does not divide the given $b \in \mathbb{Q}[s]$.

ALGORITHM 3.1. $E_b = \text{exceptionalLocusB}(f, b)$

Input: a polynomial $f \in K[\mathbf{x}]$, a polynomial $b \in \mathbb{Q}[s]$.

Output: $E_b \subset K[\mathbf{x}]$ such that $\forall P \in \text{Spec } K[\mathbf{x}]$,

$$b_{f,P} \mid b \Leftrightarrow E_b \not\subset P.$$

$G \leftarrow$ generators of $\text{in}_{(-w,w)} I_f \cap K[\mathbf{x}, s]$, where $s = -\partial_t t$.

return $\text{exceptionalLocusCore}(G, b)$.

The following subroutine computes $K[\mathbf{x}]$ -syzygies between the elements of the form $s^i g$ of s -degree at most $\deg b$ and b itself. It returns the projection of the syzygies onto the component corresponding to b .

ALGORITHM 3.2. $E_b = \text{exceptionalLocusCore}(f, b)$

Input: $G \subset K[\mathbf{x}, s]$, a polynomial $b \in \mathbb{Q}[s]$.

Output: $E_b \subset K[\mathbf{x}]$.

$G_1 \leftarrow \{a \text{ Gröbner basis of } \langle G \rangle$

w.r.t. a monomial order eliminating $s\}$.

$d \leftarrow \deg b$.

$G_2 \leftarrow \{s^i g \mid g \in G_1, i + \deg_s g \leq d\}$.

$S \leftarrow \ker \phi$ where

$$\phi : K[\mathbf{x}]^{|G_2|+1} \rightarrow \bigoplus_{i=0}^d K[\mathbf{x}] s^i$$

maps e_i , for $i = 1, \dots, |G_2|$, to the elements of G_2 and $e_{|G_2|+1}$ to b .

return projection of $S \subset K[\mathbf{x}]^{|G_2|+1}$ onto the last coordinate.

The computation of syzygies in line 4 and projection in line 5 of Algorithm 3.2 may be combined within one efficient Gröbner basis computation.

PROOF OF CORRECTNESS OF ALGORITHMS 3.1 AND 3.2. The local Bernstein–Sato polynomial $b_{f,P}$ at $P \in \text{Spec } K[\mathbf{x}]$

divides the given $b \in \mathbb{Q}[s]$ if and only if

$$\begin{aligned} Q' f^{s+1} &= b_{f,P} f^s, \text{ for some } Q' \in K[\mathbf{x}]_P \otimes D[s] \\ \Leftrightarrow Q f^{s+1} &= h b f^s, \text{ for some } Q \in D[s], h \in K[\mathbf{x}] \setminus P. \end{aligned}$$

For $h \in K[\mathbf{x}]$,

$$\begin{aligned} Q f^{s+1} &= h b f^s, \text{ for some } Q \in D[s] \\ \Leftrightarrow h b &\in \text{in}_{(-w,w)} I_f \cap K[\mathbf{x}, s] \quad (\text{by Theorem 2.2}) \\ \Leftrightarrow h &\text{ is the last coordinate of a syzygy} \\ &\text{ in the module produced by line 4} \\ \Leftrightarrow h &\in E_b. \end{aligned}$$

This proves that $b_{f,P} \mid b \Leftrightarrow E_b \not\subset P$. $\square$

REMARK 3.3. [Particulars of Algorithm 3.1] In order to compute generators of $\text{in}_{(-w,w)} I_f$, one may apply the homogenized Weyl algebra technique (for example, see [28, Algorithm 1.2.5]). Then to compute generators of $\text{in}_{(-w,w)} I_f \cap K[\mathbf{x}]\langle t, \partial_t \rangle$, eliminate $\partial_{\mathbf{x}}$ and apply the map ψ defined as follows: for a $(-w, w)$ -homogeneous $h \in K[\mathbf{x}]\langle t, \partial_t \rangle$ with $\deg_{(-w,w)} h = d$,

$$\psi(h) = \begin{cases} t^d h, & \text{if } d \geq 0, \\ \partial_t^{-d} h, & \text{if } d < 0. \end{cases}$$

This is the most expensive step of the algorithm.

We are now prepared to compute the local Bernstein–Sato polynomial of f at a prime ideal $P \subset K[\mathbf{x}]$. Its correctness follows from that of its subroutine, Algorithm 3.1.

ALGORITHM 3.4. $b = \text{localBFunction}(f, P)$

Input: a polynomial $f \in K[\mathbf{x}]$, a prime ideal $P \subset K[\mathbf{x}]$.

Output: $b \in \mathbb{Q}[s]$, the local Bernstein–Sato polynomial of f at P .

$b \leftarrow b_f$. {global b -function}

for $r \in b_f^{-1}(0)$ **do**

while $(s-r) \mid b$ **do**

$b' \leftarrow b/(s-r)$.

if $\text{exceptionalLocusB}(f, b') \subset P$ **then**

break the while loop.

else

$b \leftarrow b'$.

end if

end while

end for

return b .

REMARK 3.5. Algorithm 3.1 can also be used to compute the stratification of $\text{Spec } K[\mathbf{x}]$ according to local b -function. Below are the key steps in this procedure.

1. Compute the global b -function b_f .

2. For all roots $c \in b_f^{-1}(0)$ compute

$$E_{c,i} = \text{exceptionalLocusB}(b_f/(s-c)^{\mu_c-i}),$$

where $i \geq 0$ and is at most the multiplicity μ_c of the root c in b_f .

3. The stratum of $b = \prod_{c \in b_f^{-1}(0)} (s-c)^{i_c}$, a divisor of b_f , is

$$V \left(\bigcap_{c \in b_f^{-1}(0), i_c > 0} E_{c, i_c-1} \right) \setminus \left(\bigcup_{c \in b_f^{-1}(0)} V(E_{c, i_c}) \right).$$

This approach is similar to that in the recent work [21] of Nishiyama and Noro, which offers a more detailed treatment.

4. Generalized Bernstein–Sato polynomials

4.1 Definitions

For polynomials $\mathbf{f} = f_1, \dots, f_r \in K[\mathbf{x}]$, let $\mathbf{f}^s = \prod_{i=1}^r f_i^{s_i}$ and $Y = K^n \times K^r$ with coordinates $(\mathbf{x}, \mathbf{t})$. Define an action of $D_Y = K\langle \mathbf{x}, \mathbf{t}, \partial_{\mathbf{x}}, \partial_{\mathbf{t}} \rangle$ on $N_{\mathbf{f}} := K[\mathbf{x}][\mathbf{f}^{-1}, \mathbf{s}]\mathbf{f}^s$ as follows: x_i and ∂_{x_i} , for $i = 1, \dots, n$, act naturally and

$$\begin{aligned} t_j \cdot h(\mathbf{x}, s_1, \dots, s_j, \dots, s_r) \mathbf{f}^s &= h(\mathbf{x}, s_1, \dots, s_j + 1, \dots, s_r) f_j \mathbf{f}^s, \\ \partial_{t_j} \cdot h(\mathbf{x}, s_1, \dots, s_j, \dots, s_r) \mathbf{f}^s &= -s_j h(\mathbf{x}, s_1, \dots, s_j - 1, \dots, s_r) f_j^{-1} \mathbf{f}^s, \end{aligned}$$

for $j = 1, \dots, r$ and $h \in K[\mathbf{x}][\mathbf{f}^{-1}, \mathbf{s}]$.

With $\sigma = -(\sum_{i=1}^r \partial_{t_i} t_i)$, the *generalized Bernstein–Sato polynomial* $b_{\mathbf{f},g}$ of $\mathbf{f}$ at $g \in K[\mathbf{x}]$ is the monic polynomial $b \in \mathbb{C}[\mathbf{s}]$ of the lowest degree for which there exist $P_k \in D_X \langle \partial_{t_i} t_j \mid 1 \leq i, j \leq r \rangle$ for $k = 1, \dots, r$ such that

$$b(\sigma)g\mathbf{f}^s = \sum_{k=1}^r P_k g f_k \mathbf{f}^s. \quad (4.1)$$

REMARK 4.1. When $r = 1$, the generalized Bernstein–Sato polynomial $b_{\mathbf{f},1} = b_{\mathbf{f}}$ is the global Bernstein–Sato polynomial of $\mathbf{f} = f_1$ discussed in Section 2.

There is again an equivalent definition of $b_{\mathbf{f},g}$ by way of the V -filtration. To state this, let $V^\bullet D_Y$ denote the V -filtration of D_Y along X , where $V^m D_Y$ is D_X -generated by the set $\{t^\mu \partial_t^\nu \mid |\mu| - |\nu| \geq m\}$. The following statement may be taken as the definition of the V -filtration on $K[\mathbf{x}]$.

THEOREM 4.2. [6, Theorem 1] For $c \in \mathbb{Q}$ and sufficiently small $\epsilon > 0$, $\mathcal{J}(\mathbf{f}^c) = V^{c+\epsilon} K[\mathbf{x}]$ and $V^c K[\mathbf{x}] = \mathcal{J}(\mathbf{f}^{c-\epsilon})$.

Consider the graph of $\mathbf{f}$, which is the map $i_{\mathbf{f}} : X \rightarrow Y$ defined by $i_{\mathbf{f}}(\mathbf{x}) = (\mathbf{x}, f_1(\mathbf{x}), \dots, f_r(\mathbf{x}))$. We denote the D -module direct image of $K[\mathbf{x}]$ along $i_{\mathbf{f}}$ by

$$M_{\mathbf{f}} := (i_{\mathbf{f}})_+ K[\mathbf{x}] \cong K[\mathbf{x}] \otimes_K K\langle \partial_{\mathbf{t}} \rangle. \quad (4.2)$$

This module carries a D_Y -module structure, where the action of a vector field ξ on X and that of t_j are given by

$$\xi(p \otimes \partial_t^\nu) = \xi p \otimes \partial_t^\nu - \sum_{i=1}^r (\xi f_i) p \otimes \partial_t^{\nu+e_j}$$

$$\text{and } t_j \cdot (p \otimes \partial_t^\nu) = f_j p \otimes \partial_t^\nu - \nu_j p \otimes \partial_t^{\nu-e_j},$$

where $\partial_t^\nu = \prod_{i=1}^r \partial_{t_i}^{\nu_i}$ for $\nu = (\nu_1, \dots, \nu_r) \in \mathbb{N}$ and e_j is the element of $\mathbb{N}^r$ with j -th component equal to 1 and all others equal to 0.

Further, $M_{\mathbf{f}}$ admits a V -filtration with

$$V^m M_{\mathbf{f}} = \sum_{\nu \in \mathbb{N}^r} (V^{m+|\nu|} K[\mathbf{x}]) \otimes \partial_t^\nu.$$

For a polynomial $g \in K[\mathbf{x}]$ so that $g \otimes 1 \in M_{\mathbf{f}}$, $b_{\mathbf{f},g}$ is equal to the monic minimal polynomial of the action of σ on

$$\overline{M}_{\mathbf{f},g} := \frac{(V^0 D_Y)(g \otimes 1)}{(V^1 D_Y)(g \otimes 1)}.$$

REMARK 4.3. There is a canonical embedding of $M_{\mathbf{f}}$ into $N_{\mathbf{f}}$, where s_i is identified with $-\partial_{t_i} t_i$. In particular, for a natural number m , the image of $(V^m D_Y)(1 \otimes 1)$ under this embedding is contained in $(V^0 D_Y)\langle \mathbf{f} \rangle^m \mathbf{f}^s \subseteq N_{\mathbf{f}}$.

4.2 Algorithms

To compute the generalized Bernstein–Sato polynomial, we define the left D_Y -ideal

$$I_{\mathbf{f}} = \langle t_i - f_i \mid 1 \leq i \leq r \rangle + \langle \partial_{x_j} + \sum_{i=1}^r \frac{\partial f_i}{\partial t_j} \partial_{x_i} \mid 1 \leq j \leq n \rangle$$

that appears in the following multivariate analog of Theorem 2.1. Recall that $\sigma = -(\sum_{i=1}^r \partial_{t_i} t_i)$.

THEOREM 4.4. The ideal $I_{\mathbf{f}}$ is equal to $\text{Ann}_{D_Y} \mathbf{f}^s$. Furthermore, the ideal $\text{Ann}_{D_X[\mathbf{s}]} \mathbf{f}^s$ equals the image of $I_{\mathbf{f}} \cap D_X[\sigma]$ under the substitution $\sigma \mapsto s$.

We now provide two subroutines in our computations of Bernstein–Sato polynomials and multiplier ideals. The first finds the left side of a functional equation of the form (4.1) without an expensive elimination step. The second finds the homogenization of a D_Y -ideal with respect to the weight vector $(-w, w)$, where $w = (\mathbf{0}, \mathbf{1}) \in \mathbb{R}^n \times \mathbb{R}^r$ determines an elimination term order for X in Y .

ALGORITHM 4.5. $b = \text{linearAlgebraTrick}(g, G)$

Input: generators G of an ideal $I \subset D_Y$,
a polynomial $g \in K[\mathbf{x}]$,
such that there is $b \in K[\mathbf{s}]$ with $b(\sigma)g \in I$.
Output: b , the monic polynomial of minimal degree such that $b(\sigma)g \in I$.
 $B \leftarrow \{a \text{ Gröbner basis of } D_Y G\}$.
 $d \leftarrow 0$.
repeat
 $d \leftarrow d + 1$
until $\exists (c_0, \dots, c_d) \in K^{d+1}$ such that $c_d = 1$ and

$$\sum_{i=0}^d c_i \text{NF}_B(\sigma^i g) = 0.$$

return $\sum_{i=0}^d c_i s^i$.

ALGORITHM 4.6. $G^* = \text{starIdeal}(G, w)$

Input: generators G of an ideal $J \subset D_Y$,
a weight vector $w \in \mathbb{Z}^{n+r}$.
Output: $G^* \subset \text{gr}_{(-w,w)} D_Y \cong D_Y$, a set of generators of the ideal J^* of $(-w, w)$ -homogeneous elements of J .
 $G^h \leftarrow$ generators G homogenized w.r.t. weight $(-w, w)$;
 $G^h \subset D_Y[h]$ with a homogenizing variable h of weight 1.
 $B \leftarrow \{a \text{ Gröbner basis of } (G^h, hu - 1) \subset D_Y[h, u]\}$
w.r.t. a monomial order eliminating $\{h, u\}$.
return $B \cap D_Y$.

Below are two algorithms that are simplified versions of Shibuta's algorithms for the generalized Bernstein–Sato polynomial. In the first, we use a module $D_Y[\mathbf{s}]$, where the new variable s commutes with all variables in D_Y .

ALGORITHM 4.7. $b_{\mathbf{f},g} = \text{generalB}(\mathbf{f}, g, \text{StarIdeal})$

Input: $\mathbf{f} = \{f_1, \dots, f_r\} \subset K[\mathbf{x}]$, $g \in K[\mathbf{x}]$.
Output: $b_{\mathbf{f},g}$, the generalized Bernstein–Sato polynomial of $\mathbf{f}$ at g .
 $G_1 \leftarrow \{t_j - f_j \mid j = 1, \dots, r\}$
 $\cup \{\partial_{x_i} + \sum_{j=1}^r \frac{\partial f_j}{\partial x_i} \partial_{t_j} \mid i = 1, \dots, n\}$.

$G_2 \leftarrow \text{starIdeal}(G_1, w)$
 $\cup \langle gf_i \mid 1 \leq i \leq r \rangle \cup \{s - \sigma\} \subset D_Y[s],$
where w assigns weight 1 to all ∂_{t_j} and 0 to all ∂_{x_i} .
return linearAlgebraTrick(G_2).

ALGORITHM 4.8. $b_{f,g} = \text{generalB}(\mathbf{f}, g, \text{InitialIdeal})$

Input: $\mathbf{f} = \{f_1, \dots, f_r\} \subset K[\mathbf{x}]$, $g \in K[\mathbf{x}]$.

Output: $b_{f,g}$, the generalized Bernstein–Sato polynomial of $\mathbf{f}$ at g .

$G_1 \leftarrow \{t_j - f_j \mid j = 1, \dots, r\}$
 $\cup \{\partial_{x_i} + \sum_{j=1}^r \frac{\partial f_j}{\partial x_i} \partial_{t_j} \mid i = 1, \dots, n\}.$

$G_2 \leftarrow G_1 \cap D_Y \cdot g.$

$G_3 \leftarrow \text{generators of } \text{in}_{(-w,w)}(G_2) \text{ where } w \text{ assigns weight 1 to all } \partial_{t_j} \text{ and 0 to all } \partial_{x_i}.$

return linearAlgebraTrick(G_3).

Their correctness follows from [32, Theorems 3.4 and 3.5].

REMARK 4.9. According to the experiments in [15] a modification of Algorithm 4.6 that uses elimination involving one less additional variable exhibits better performance. Our current implementation does not take advantage of this.

4.3 Applications

The study of the generalized Bernstein–Sato polynomial in [6] yields several applications of our algorithms, which we mention here. Each has been implemented in Macaulay2.

We begin with a result that shows that comparison with the roots of $b_{f,g}(s)$ provides a membership test for $\mathcal{J}(\mathbf{f}^c)$ for any positive rational number c .

PROPOSITION 4.10. [6, Corollary 2] Let $g \in K[\mathbf{x}]$ and fix a positive rational number c . Then $g \in \mathcal{J}(\mathbf{f}^c)$ if and only if c is strictly less than all roots of $b_{f,g}(-s)$.

When $\mathbf{f}$ defines a complete intersection, Algorithms 4.7 and 4.8 provide tests to determine if Z has at most rational singularities.

THEOREM 4.11. [6, Theorem 4] Suppose that Z is a complete intersection of codimension r in Y defined by $\mathbf{f} = f_1, \dots, f_r$. Then Z has at most rational singularities if and only if $\text{lct}(\mathbf{f}) = r$ and has multiplicity one as a root of $b_{\mathbf{f}}(-s)$.

To compute a local version of the generalized Bernstein–Sato polynomial, we need the following analog of Theorem 2.2.

THEOREM 4.12. Let $b(\mathbf{x}, s)$ be a nonzero polynomial in $K[\mathbf{x}, s]$. Then $b(\mathbf{x}, \sigma) \in \text{in}_{(-w,w)} I_{\mathbf{f}} \cap K[\mathbf{x}, \sigma]$ if and only if there exist $Q_k \in D[s]$ s.t. $\sum_{k=1}^r Q_k f_k \mathbf{f}^s = b(\mathbf{x}, s) \mathbf{f}^s$.

PROOF. This follows by the same argument as that of Theorem 2.2. $\square$

REMARK 4.13. In light of Theorem 4.12, the strategy in Section 3 yields a computation of the local version of the generalized Bernstein–Sato polynomial. The only significant difference comes from the lack of an analogue to the map ψ of Remark 3.3. However, it is still possible to compute $\text{in}_{(-w,w)} I_{\mathbf{f}} \cap K[\mathbf{x}, \sigma]$ by adjoining one more variable s to the algebra and $s - \sigma$ to the ideal and eliminating $\mathbf{t}$ and $\partial_{\mathbf{t}}$. In case of the hypersurface this is a more expensive strategy than the one described in Remark 3.3.

5. Multiplier ideals via m -generalized Bernstein–Sato polynomials

For this section, we retain the notation of Section 4 and discuss Shibuta’s m -generalized Bernstein–Sato polynomials. These are defined using the V -filtration of D_Y along X , but they also possess an equational definition. In contrast to the generalized Bernstein–Sato polynomials of Section 4, this generalization allows us to simultaneously consider families of polynomials $K[\mathbf{x}]$, yielding a method to compute multiplier ideals.

DEFINITION 5.1. Let $\overline{M}_{\mathbf{f}}^{(m)} = (V^0 D_Y) \delta / (V^m D_Y) \delta$ with $\delta = 1 \otimes 1 \in M_{\mathbf{f}} \cong K[\mathbf{x}] \otimes_K K\langle \partial_{\mathbf{t}} \rangle$. Define the m -generalized Bernstein–Sato polynomial $b_{f,g}^{(m)}$ to be the monic minimal polynomial of the action of $\sigma := -(\sum_{i=1}^r \partial_{t_i} t_i)$ on

$$\overline{M}_{f,g}^{(m)} := (V^0 D_Y) g \overline{\otimes 1} \subseteq \overline{M}_{\mathbf{f}}^{(m)}.$$

REMARK 5.2. Since $M_{\mathbf{f}}$ is V -filtered, the polynomial $b_{f,g}^{(m)}$ is nonzero and its roots are rational.

PROPOSITION 5.3. The m -generalized $b_{f,g}^{(m)}$ is equal to the monic polynomial $b(s)$ of minimal degree in $K[s]$ such that there exist $P_k \in D_X \langle -\partial_{t_i} t_j \mid 1 \leq i, j \leq r \rangle$ and $h_k \in \langle \mathbf{f} \rangle^m$ such that in $N_{\mathbf{f}}$ there is an equality

$$b(\sigma) g \mathbf{f}^s = \sum_{k=1}^r P_k h_k \mathbf{f}^s. \quad (5.1)$$

PROOF. By the embedding in Remark 4.3, the existence of such an equation is equivalent to the existence of $Q_k \in D_X \langle -\partial_{t_i} t_j \mid 1 \leq i, j \leq r \rangle$ and $\mu(k) \in \mathbb{N}^r$ with $|\mu(k)| \geq m$ such that in $M_{\mathbf{f}}$, $b(\sigma) \cdot (g \otimes 1) = \sum_{k=1}^r Q_k \mathbf{t}^{\mu(k)} \cdot (1 \otimes 1)$. $\square$

REMARK 5.4. Since $(V^0 D_Y) g \overline{\otimes 1} \subseteq \overline{M}_{\mathbf{f}}^{(1)}$ is a quotient of $\overline{M}_{f,g}$, the generalized Bernstein–Sato polynomial $b_{f,g}$ is a multiple of the m -generalized Bernstein–Sato polynomial $b_{f,g}^{(1)}$. When g is a unit, the equality $b_{f,g} = b_{f,g}^{(1)}$ holds, as seen easily by comparing (4.1) and (5.1). However, this equality does not hold in general.

EXAMPLE 5.5. When $n = 3$ and $\mathbf{f} = \sum_{i=1}^3 x_i^2$, we have

$$b_{f,x_1}(s) = (s+1)(s+\frac{5}{2}) \quad \text{and} \quad b_{f,x_1}^{(1)}(s) = s+1.$$

In particular, $b_{f,x_1}^{(1)}$ strictly divides b_{f,x_1} .

Proposition 5.3 translates into the following algorithm.

ALGORITHM 5.6. $b_{f,g}^{(m)} = \text{generalB}(\mathbf{f}, g, m)$

Input: $\mathbf{f} = \{f_1, \dots, f_r\} \subset K[\mathbf{x}]$, $g \in K[\mathbf{x}]$, $m \in \mathbb{Z}_{>0}$.

Output: $b_{f,g}^{(m)}$, the m -generalized Bernstein–Sato polynomial as defined in Definition 5.1.

$G_1 \leftarrow \{t_j - f_j \mid j = 1, \dots, r\}$

$\cup \{\partial_{x_i} + \sum_{j=1}^r \frac{\partial f_j}{\partial x_i} \partial_{t_j} \mid i = 1, \dots, n\}.$

$G_2 \leftarrow \text{starIdeal}(G_1, w) \cup \{\mathbf{f}^\alpha \mid \alpha \in \mathbb{N}^r, |\alpha| = m\},$
where w assigns weight 1 to all ∂_{t_j} and 0 to all ∂_{x_i} .

return linearAlgebraTrick(g, G_2).

5.1 Jumping coefficients and the log canonical threshold

For the remainder of this article, set $K = \mathbb{C}$. Our algorithms for multiplier ideals are motivated by the following result.

THEOREM 5.7. [32, Theorem 4.3] For $g \in K[\mathbf{x}]$ and $c < m + \text{lt}(\mathbf{f})$, $g \in \mathcal{J}(\mathbf{f}^c)$ if and only if c is strictly less than every root of $b_{\mathbf{f},g}^{(m)}(-s)$. In other words,

$$\mathcal{J}(\mathbf{f}^c) = \{g \in K[\mathbf{x}] \mid b_{\mathbf{f},g}^{(m)}(-\alpha) = 0 \Rightarrow c < \alpha\}.$$

PROOF. By Theorem 4.2, $\mathcal{J}(\mathbf{f}^c) = V^{c+\epsilon}K[\mathbf{x}]$ for all sufficiently small $\epsilon > 0$. Hence, $g \in \mathcal{J}(\mathbf{f}^c)$ precisely when $g \otimes 1 \in V^\alpha M_{\mathbf{f}}$ for all $\alpha \leq c$, or equivalently,

$$c < \max\{\alpha \mid g \otimes 1 \in V^\alpha M_{\mathbf{f}}\}. \quad (5.2)$$

As in [6, (2.3.1)], the right side of (5.2) is equal to $\min\{\alpha \mid \text{Gr}_V^\alpha((V^0 D_Y)(g \otimes 1)) \neq 0\}$ and strictly less than $\min\{\alpha \mid \text{Gr}_V^\alpha((V^m D_Y)\delta) \neq 0\}$. Thus by our choice of m , $g \in \mathcal{J}(\mathbf{f}^c)$ exactly when c is strictly greater than $\min\{\alpha \mid \text{Gr}_V^\alpha \overline{M}_{\mathbf{f},g}^{(m)} \neq 0\}$. The theorem now follows because $\text{Gr}_V^\alpha \overline{M}_{\mathbf{f},g}^{(m)} \neq 0$ if and only if $b_{\mathbf{f},g}^{(m)}(-\alpha) = 0$. $\square$

Theorem 5.7 provides a second membership test for membership in $\mathcal{J}(\mathbf{f}^c)$; moreover, the following corollary provides a method for computing the log canonical threshold and jumping coefficients of $\mathbf{f}$ via the m -generalized Bernstein–Sato polynomial $b_{\mathbf{f}}^{(1)} = b_{\mathbf{f},1}^{(1)}$.

COROLLARY 5.8. For any positive integer m , the minimal root of $b_{\mathbf{f}}^{(m)}(-s)$ is equal to the log-canonical threshold $\text{lt}(\mathbf{f})$ of $\langle \mathbf{f} \rangle \subseteq K[\mathbf{x}]$. Further, the jumping coefficients of $\langle \mathbf{f} \rangle$ within the interval $[\text{lt}(\mathbf{f}), \text{lt}(\mathbf{f}) + m]$ are all roots of $b_{\mathbf{f}}^{(m)}(-s)$.

5.2 Computing multiplier ideals

Here we present an algorithm to compute multiplier ideals that simplifies the method of Shibuta [32]. In particular, significant improvement is achieved bypassing the primary decomposition computations required by Shibuta’s method.

For a positive integer m , define the $K[\mathbf{x}, \sigma]$ -ideal

$$J_{\mathbf{f}}(m) = (I_{\mathbf{f}}^* + D_Y \cdot \langle \mathbf{f} \rangle^m) \cap K[\mathbf{x}, \sigma],$$

where $I_{\mathbf{f}}^* \subset D_Y$ is the ideal of the $(-w, w)$ -homogeneous elements of $I_{\mathbf{f}}$. This ideal is closely related to the m -generalized Bernstein–Sato polynomials.

LEMMA 5.9. For $g \in K[\mathbf{x}]$, the m -generalized Bernstein–Sato polynomial $b_{\mathbf{f},g}^{(m)}$ is equal to the monic polynomial $b(s) \in K[s]$ of minimal degree such that

$$\langle b(\sigma) \rangle = (J_{\mathbf{f}}(m) : g) \cap K[\sigma]. \quad (5.3)$$

PROOF. By (5.1), $b_{\mathbf{f},g}^{(m)}$ is the monic polynomial $b(s) \in K[s]$ of minimal degree such that

$$b(\sigma)g \in I_{\mathbf{f}} + D_X \langle -\partial_{t_i} t_j \mid 1 \leq i, j \leq r \rangle \cdot \langle \mathbf{f} \rangle^m.$$

Since $b(\sigma)g$ is $(-w, w)$ -homogeneous, we obtain (5.3). $\square$

THEOREM 5.10. [32, Theorem 4.4] Let $J_{\mathbf{f}}(m) = \bigcap_{i=1}^l q_i$ be a primary decomposition with $q_i \cap K[\sigma] = (\sigma + c(i))^{\kappa(i)}$ for some positive integer $\kappa(i)$. Then for $c < \text{lt}(\mathbf{f}) + m$,

$$\mathcal{J}(\mathbf{f}^c) = \bigcap_{j: c(j) \geq c} (q_j \cap K[\mathbf{x}]).$$

PROOF. We see from (5.1) that $b_{\mathbf{f},g}^{(m)}(s)$ is the monic polynomial $b(s)$ of minimal degree such that there exist some

$P_k \in D_X \langle -\partial_{t_i} t_j \mid 1 \leq i, j \leq r \rangle$ and $h_k \in \langle \mathbf{f} \rangle^m$ such that $(b(\sigma)g - \sum_k P_k h_k) \in I_{\mathbf{f}}$. Equivalently,

$$b(\sigma)g \in (I_{\mathbf{f}}^* + D_Y \cdot \langle \mathbf{f} \rangle^m) \cap K[\mathbf{x}, \sigma].$$

The theorem now follows from Lemma 5.9. $\square$

The following is based on methodology used in the computation of the local b -function and, in particular, employs Algorithm 3.2. Its correctness follows immediately from Theorem 5.10 and the results of Section 3.

ALGORITHM 5.11. $\mathcal{J}(\mathbf{f}^c) = \text{multiplierIdeal}(\mathbf{f}, c)$

Input: $\mathbf{f} = \{f_1, \dots, f_r\} \subset K[\mathbf{x}]$, $c \in \mathbb{Q}$.

Output: $\mathcal{J}(\mathbf{f}^c)$, the multiplier ideal of $\mathbf{f}$ with coefficient c .

$G_1 \leftarrow \{t_j - f_j \mid j = 1, \dots, r\}$

$\cup \{\partial_{x_i} + \sum_{j=1}^r \frac{\partial f_j}{\partial x_i} \partial_{t_j} \mid i = 1, \dots, n\}.$

$m \leftarrow \lceil \max\{c - \text{lt}(\mathbf{f}), 1\} \rceil.$

if $c - \text{lt}(\mathbf{f})$ is integer and ≥ 1 **then**

$m \leftarrow m + 1$

end if

$G_2 \leftarrow \text{starIdeal}(G_1, w) \cup \{\mathbf{f}^\alpha \mid \alpha \in \mathbb{N}^r, |\alpha| = m\}$
 $\cup \{s - \sigma\} \subset D_Y[s],$

where w assigns weight 1 to all ∂_{t_j} and 0 to all ∂_{x_i} .

$B \leftarrow \left\{ \begin{array}{l} \text{a Gröbner basis of } G_2 \text{ w.r.t.} \\ \text{an order eliminating } \{\partial_{\mathbf{x}}, \mathbf{t}, \partial_{\mathbf{t}}\} \end{array} \right\} \cap K[\mathbf{x}, s].$

$b \leftarrow \text{generalB}(\mathbf{f}, 1, m).$

$\{ \text{The computation of } b_{\mathbf{f},1}^{(m)} \text{ may make use of } B. \}$

$b' \leftarrow \text{product of factors } (s - c')^{\alpha(c')} \text{ of } b \text{ over all roots } c' \text{ of } b_{\mathbf{f}}^{(m)} \text{ such that } -c' > c, \text{ where } \alpha(c') \text{ equals the multiplicity of the root } c'.$

return $\text{exceptionalLocusCore}(B, b').$

As noted in [32, Remark 4.6.ii], for a nonnegative rational number c , $\mathcal{J}(\mathbf{f}^c) = \langle \mathbf{f} \rangle \cdot \mathcal{J}(\mathbf{f}^{c-1})$ when c is at least equal to the analytic spread $\lambda(\mathbf{f})$ of $\langle \mathbf{f} \rangle$. (The analytic spread of $\langle \mathbf{f} \rangle$ is the least number of generators for an ideal I such that $\langle \mathbf{f} \rangle$ is integral over I .) Hence, to find generators for any multiplier ideal of $\mathbf{f}$, it is enough to compute $J_{\mathbf{f}}(m)$ for one $m \geq \lambda(\mathbf{f}) - \text{lt}(\mathbf{f})$.

When it is known that the multiplier ideal $\mathcal{J}(\mathbf{f}^c)$ is 0-dimensional, it is possible to bypass the elimination step (line 7 of Algorithm 5.11) in the following fashion. For a fixed monomial ordering $\geq$ on $K[\mathbf{x}]$, we know that there are finitely many standard monomials (monomials not in the initial ideal $\text{in}_{\geq} \mathcal{J}(\mathbf{f}^c)$). Let $b' \in \mathbb{Q}[s]$ be the polynomial produced by lines 8 and 9 of the above algorithm. A basis for the K -linear relations amongst $\{\mathbf{x}^\alpha b'(\sigma) \mid |\alpha| \leq d\}$ modulo $J_{\mathbf{f}}(m)$ gives a basis P_d for the K -space of polynomials in $\mathcal{J}(\mathbf{f}^c)$ up to degree d . By starting with $d = 0$ and incrementing d until all monomials of degree d belong to $\text{in}_{\geq} \langle P_{d-1} \rangle$, we obtain $\langle P_d \rangle = \mathcal{J}(\mathbf{f}^c)$ upon termination.

ALGORITHM 5.12. $\mathcal{J}(\mathbf{f}^c) = \text{multiplierIdealLA}(\mathbf{f}, c, d_{\max})$

Input: $\mathbf{f} = \{f_1, \dots, f_r\} \subset K[\mathbf{x}]$, $c \in \mathbb{Q}$, $d_{\max} \in \mathbb{N}$.

Output: the multiplier ideal $\mathcal{J}(\mathbf{f}^c) \subset K[\mathbf{x}]$, in case it is generated in degrees at most $d_{\max}$.

$G_1 \leftarrow \{t_j - f_j \mid j = 1, \dots, r\}$

$\cup \{\partial_{x_i} + \sum_{j=1}^r \frac{\partial f_j}{\partial x_i} \partial_{t_j} \mid i = 1, \dots, n\}.$

$m \leftarrow \lceil \max\{c - \text{lt}(\mathbf{f}), 1\} \rceil.$

if $c - \text{lt}(\mathbf{f})$ is integer and ≥ 1 **then**

$m \leftarrow m + 1$

end if

$G_2 \leftarrow \text{starIdeal}(G_1, w) \cup \{\mathbf{f}^\alpha \mid \alpha \in \mathbb{N}^r, |\alpha| = m\} \subset D_Y$,
where w assigns weight 1 to all ∂_{t_j} and 0 to all ∂_{x_i} .

$B \leftarrow \left\{ \begin{array}{l} \text{a Gröbner basis of } G_2 \text{ w.r.t.} \\ \text{any monomial order} \end{array} \right\}$.

$b \leftarrow \text{generalB}(\mathbf{f}, 1, m)$.

$b' \leftarrow$ product of factors $(s - c')^{\alpha(c')}$ of b over all roots c' of $b_f^{(m)}$ such that $-c' > c$, where $\alpha(c')$ equals the multiplicity of the root c' .

$d \leftarrow -1$; $P \leftarrow \emptyset \subset K[\mathbf{x}]$ with $\geq$ that respects degree.

while $P = \emptyset$ **or** $(\text{in}_{\geq} \langle P \rangle)$ does not contain all monomials of degree d **and** $d < d_{\max}$ **do**

$d \leftarrow d + 1$,

$A \leftarrow \{\alpha \mid |\alpha| \leq d, \mathbf{x}^\alpha \notin \text{in}_{\geq} \langle P \rangle\}$

Find a basis Q for the K -syzygies $(q_\alpha)_{\alpha \in A}$ such that

$$\sum_{\alpha \in A} q_\alpha \text{NF}_B(\mathbf{x}^\alpha b(\sigma)) = 0.$$

$P \leftarrow P \cup \{\sum_{\alpha \in A} q_\alpha \mathbf{x}^\alpha \mid (q_\alpha) \in Q\}$.

end while

return $\langle P \rangle$.

Notice that with $d_{\max} = \infty$ the algorithm terminates in case $\dim \mathcal{J}(\mathbf{f}^c) = 0$. It also can be used to provide a K -basis of the up-to-degree- $d_{\max}$ part of an ideal of any dimension.

6. Examples

We have tested our implementation on the problems in [32]. In addition, this section provides examples from other sources with the theoretically known Bernstein–Sato polynomials, log-canonical thresholds, jumping numbers, and/or multiplier ideals; below is the output of our algorithms on several of them.

The authors would like to thank Zach Teitler for suggesting interesting examples, some of which are beyond the reach of our current implementation. We also thank Takafumi Shibuta for sharing his script (written in *risa/asir* [23]), which is the only other existing software for computing multiplier ideals.

A note on how to access Macaulay2 scripts generating examples, including the ones in this paper and some unsolved challenges, is posted at [3] along with other useful links.

EXAMPLE 6.1. When $f = x^5 + y^4 + x^3y^2$, Saito observed that not all roots of $b_f(-s)$ are jumping coefficients [27, Example 4.10]. The roots of $b_f(-s)$ within the interval $(0, 1]$ are

$$\frac{9}{20}, \frac{11}{20}, \frac{13}{20}, \frac{7}{10}, \frac{17}{20}, \frac{9}{10}, \frac{19}{20}, 1.$$

However, $\frac{11}{20}$ is not a jumping coefficient of f . This can be seen in the ideal $J_f(1)$ from Theorem 5.10, which has, among others, the primary components $\langle s + \frac{9}{20}, y, x \rangle$ and $\langle s + \frac{11}{20}, y, x \rangle$. In fact,

$$\mathcal{J}(f^c) = \begin{cases} \mathbb{C}[x, y] & \text{if } 0 \leq c < \frac{9}{20}, \\ \langle x, y \rangle & \text{if } \frac{9}{20} \leq c < \frac{13}{20}, \\ \langle x^2, y \rangle & \text{if } \frac{13}{20} \leq c < \frac{7}{10}, \\ \langle x^2, xy, y^2 \rangle & \text{if } \frac{7}{10} \leq c < \frac{17}{20}, \\ \langle x^3, xy, y^2 \rangle & \text{if } \frac{17}{20} \leq c < \frac{9}{10}, \\ \langle x^3, x^2y, y^2 \rangle & \text{if } \frac{9}{10} \leq c < \frac{19}{20}, \\ \langle x^3, x^2y, xy^2, y^3 \rangle & \text{if } \frac{19}{20} \leq c < 1, \end{cases}$$

and $\mathcal{J}(f^c) = \langle \mathbf{f} \rangle \cdot \mathcal{J}(f^{c-1})$ for all $c \geq 1$.

EXAMPLE 6.2. We compute Bernstein–Sato polynomials to verify examples corresponding to [34, Example 7.1]. The $\mathbb{C}[x, y, z]$ -ideal

$$\langle \mathbf{f} \rangle = \langle x - z, y - z \rangle \cap \langle 3x - z, y - 2z \rangle \cap \langle 5y - x, z \rangle$$

defining three non-collinear points in $\mathbb{P}^2$ has

$$b_{\mathbf{f}}(s) = (s + \frac{3}{2})(s + 2)^2.$$

In particular, its log canonical threshold is $\frac{3}{2}$. The multiplier ideals in this case are

$$\mathcal{J}(f^c) = \begin{cases} \mathbb{C}[x, y, z] & \text{if } 0 \leq c < \frac{3}{2}, \\ \langle x, y, z \rangle & \text{if } \frac{3}{2} \leq c < 2, \end{cases}$$

and $\mathcal{J}(f^c) = \langle \mathbf{f} \rangle \cdot \mathcal{J}(f^{c-1})$ for all $c \geq 2$. On the other hand, the $\mathbb{C}[x, y, z]$ -ideal

$$\langle \mathbf{g} \rangle = \langle y, z \rangle \cap \langle x - 2z, y - z \rangle \cap \langle 2x - 3z, y - z \rangle$$

defines three collinear points in $\mathbb{P}^2$. Since

$$b_{\mathbf{g}}(s) = (s + \frac{5}{3})(s + 2)^2(s + \frac{7}{3}),$$

the log canonical threshold of $\mathbf{g}$ is $\frac{5}{3}$. Here the multiplier ideals are

$$\mathcal{J}(\mathbf{g}^c) = \begin{cases} \mathbb{C}[x, y, z] & \text{if } 0 \leq c < \frac{5}{3}, \\ \langle x, y, z \rangle & \text{if } \frac{5}{3} \leq c < 2, \end{cases}$$

and $\mathcal{J}(\mathbf{g}^c) = \langle \mathbf{g} \rangle \cdot \mathcal{J}(\mathbf{g}^{c-1})$ for all $c \geq 2$. Thus, as Teitler points out, although $\mathbf{g}$ defines a more special set than $\mathbf{f}$, it yields a less singular variety.

EXAMPLE 6.3. Consider $f = (x^2 - y^2)(x^2 - z^2)(y^2 - z^2)z$, the defining equation for a nongeneric hyperplane arrangement. Saito showed that $\frac{5}{7}$ is a root of $b_f(-s)$ but not a jumping coefficient [26, 5.5]. We verified this, obtaining the root 1 of $b_f(-s)$ with multiplicity 3, as well the following roots of multiplicity 1 (including $\frac{5}{7}$):

$$\frac{3}{7}, \frac{4}{7}, \frac{2}{3}, \frac{5}{7}, \frac{6}{7}, \frac{8}{7}, \frac{9}{7}, \frac{4}{3}, \frac{10}{7}, \frac{11}{7}.$$

Further,

$$\mathcal{J}(f^c) = \begin{cases} \mathbb{C}[x, y, z] & \text{if } 0 \leq c < \frac{3}{7}, \\ \langle x, y, z \rangle & \text{if } \frac{3}{7} \leq c < \frac{4}{7}, \\ \langle x, y, z \rangle^2 & \text{if } \frac{4}{7} \leq c < \frac{2}{3}, \\ \langle z, x \rangle \cap \langle z, y \rangle \cap \langle y + z, x + z \rangle \cap \langle y + z, x - z \rangle \cap \langle y - z, x + z \rangle \cap \langle y - z, x - z \rangle & \text{if } \frac{2}{3} \leq c < \frac{6}{7}, \\ \langle z, x \rangle \cap \langle z, y \rangle \cap \langle y + z, x + z \rangle \cap \langle y + z, x - z \rangle \cap \langle y - z, x + z \rangle \cap \langle y - z, x - z \rangle & \text{if } \frac{6}{7} \leq c < 1, \end{cases}$$

and $\mathcal{J}(f^c) = \langle \mathbf{f} \rangle \cdot \mathcal{J}(f^{c-1})$ for all $c \geq 1$.

All examples in this section involve multiplier ideals of low dimension. In our experience, Algorithm 5.12 for a multiplier ideal of positive yet low dimension with a large value for $d_{\max}$ runs significantly faster than Algorithm 5.11. This is due to the avoidance of an expensive elimination step.

7. REFERENCES

- [1] D. Andres, V. Levandovskyy, and J. Morales. Principal intersection and Bernstein–Sato polynomial of an affine variety. *ISSAC 2009*, 231–238. ACM, New York, 2000.
- [2] R. Bahloul and T. Oaku. Local Bernstein–Sato ideals: algorithm and examples. *J. Symbolic Comput.* 45 (2010), no. 1, 46–59.
- [3] C. Berkesch and A. Leykin. Multiplier ideals in Macaulay2. <http://people.math.gatech.edu/~aleykin3/MultiplierIdeals>.
- [4] I. N. Bernstein. Analytic continuation of generalized functions with respect to a parameter. *Functional Anal. Appl.* 6:273–285, 1972.
- [5] J. Briançon and Ph. Maisonobe. Remarques sur l’ideal de Bernstein associé à des polynômes. Preprint, 2002.
- [6] N. Budur, M. Mustață, and M. Saito. Bernstein–Sato polynomials of arbitrary varieties. *Compos. Math.* 142 (2006), no. 3, 779–797.
- [7] L. Ein, R. Lazarsfeld, K. E. Smith, and D. Varolin. Jumping coefficients of multiplier ideals. *Duke Math. J.* 123 (2004), no. 3, 469–506.
- [8] D.R. Grayson and M.E. Stillman. Macaulay 2, a software system for research in algebraic geometry. <http://www.math.uiuc.edu/Macaulay2/>.
- [9] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 2.0. A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern, 2001. <http://www.singular.uni-kl.de>.
- [10] J. Howald. Multiplier ideals of monomial ideals. *Trans. Amer. Math. Soc.* 353 (2001), no. 7, 2665–2671.
- [11] M. Kashiwara. B -functions and holonomic systems. Rationality of roots of B -functions. *Invent. Math.* 38(1):33–53, 1976/77.
- [12] M. Kashiwara. Vanishing cycle sheaves and holonomic systems of differential equations. *Algebraic geometry (Tokyo/Kyoto, 1982)*, volume 1016 of *Lecture Notes in Math.*, 134–142. Springer, Berlin, 1983.
- [13] R. Lazarsfeld. *Positivity in algebraic geometry. II. Positivity for vector bundles, and multiplier ideals*. A Series of Modern Surveys in Mathematics 49. Springer-Verlag, Berlin, 2004.
- [14] R. Lazarsfeld. A short course on multiplier ideals. Notes, 2009. [arXiv:0901.0561v1](http://arxiv.org/abs/0901.0561v1).
- [15] V. Levandovskyy and J. Martín Morales. Computational D -module theory with SINGULAR, comparison with other systems and two new algorithms. *ISSAC 2008*, 173–180. ACM, New York, 2008.
- [16] A. Leykin and H. Tsai. Software package “ D -modules for Macaulay2”. <http://people.math.gatech.edu/~aleykin3/Dmodules>.
- [17] B. Malgrange. Le polynôme de Bernstein d’une singularité isolée. (French). *Fourier integral operators and partial differential equations (Colloq. Internat., Univ. Nice, Nice, 1974)*, volume 459 of *Lecture Notes in Math.* 98–119. Springer, Berlin, 1975.
- [18] B. Malgrange. Polynômes de Bernstein–Sato et cohomologie évanescence. *Analysis and topology on singular spaces, II, III (Luminy, 1981)*, volume 101 of *Astérisque*, 243–267. Soc. Math. France, Paris, 1983.
- [19] M. Mustață. Multiplier ideals of hyperplane arrangements. *Trans. Amer. Math. Soc.* 358 (2006), no. 11, 5015–5023.
- [20] H. Nakayama. Algorithm computing the local b -function by an approximate division algorithm in $\hat{D}$. *J. Symbolic Comput.* 44(5):449–462, 2009.
- [21] K. Nishiyama and M. Noro. Stratification associated with local b -function. *J. Symbolic Comput.* 45(4):462–480, 2010.
- [22] M. Noro. An efficient modular algorithm for computing the global b -function. *Mathematical Software: ICMS 2002, World Sci. Publ.*, 147–157, 2002.
- [23] M. Noro, T. Shimoyama, and T. Takeshima. Computer algebra system risa/asir. <http://www.math.kobe-u.ac.jp/Asir/index.html>.
- [24] T. Oaku. Algorithms for the b -function and D -modules associated with a polynomial. *J. Pure Appl. Algebra*, 117/118:495–518, 1997. Algorithms for algebra (Eindhoven, 1996).
- [25] M. Saito. Introduction to a theory of b -functions. Preprint, 2006. [arXiv:math/0610783v1](http://arxiv.org/abs/math/0610783v1).
- [26] M. Saito. Multiplier ideals, b -function, and spectrum of a hypersurface singularity. *Compos. Math.* 143 (2007), no. 4, 1050–1068.
- [27] M. Saito. On b -function, spectrum and multiplier ideals. Algebraic analysis and around, 355–379, *Adv. Stud. Pure Math.* 54, Math. Soc. Japan, Tokyo, 2009.
- [28] M. Saito, B. Sturmfels, and N. Takayama. *Gröbner deformations of hypergeometric differential equations*, volume 6 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 2000.
- [29] M. Saito and T. Shintani. On zeta functions associated with prehomogeneous vector spaces. *Proc. Nat. Acad. Sci. U.S.A.*, 69:1081–1082, 1972.
- [30] M. Schulze. The differential structure of the Brieskorn lattice. *A.M. Cohen et al.: Mathematical Software - ICMS 2002*. World Sci. Publ., 2002.
- [31] M. Schulze. A normal form algorithm for the Brieskorn lattice. *J. Symbolic Comput.* 38, 4 (2004), 1207–1225.
- [32] T. Shibuta. An algorithm for computing multiplier ideals. Preprint, 2010. [arXiv:0807.4302v6](http://arxiv.org/abs/0807.4302v6).
- [33] N. Takayama. **kan/sml**: a computer algebra system for algebraic analysis. www.math.sci.kobe-u.ac.jp/KAN/.
- [34] Z. Teitler. Multiplier ideals of general line arrangements in $\mathbb{C}^3$. *Comm. Algebra*, 35 (2007), no. 6, 1902–1913.
- [35] Z. Teitler. A note on Mustață’s computation of multiplier ideals of hyperplane arrangements. *Proc. Amer. Math. Soc.* 136 (2008), no. 5, 1575–1579.
- [36] J.M. Ucha and F.J. Castro-Jiménez. Bernstein–Sato ideals associated to polynomials. *J. Symbolic Comput.* 37(5):629–639, 2004.

Global Optimization of Polynomials Using Generalized Critical Values and Sums of Squares *

Feng Guo
Key Laboratory of
Mathematics Mechanization,
AMSS
Beijing 100190, China
fguo@mmrc.iss.ac.cn

Mohab Safey El Din
UPMC, Univ Paris 06, INRIA,
Paris-Rocquencourt Center,
SALSA Project, LIP6/CNRS
UMR 7606, France
Mohab.Safey@lip6.fr

Lihong Zhi
Key Laboratory of
Mathematics Mechanization,
AMSS
Beijing 100190, China
lzhi@mmrc.iss.ac.cn

ABSTRACT

Let $\bar{X} = [X_1, \dots, X_n]$ and $f \in \mathbb{R}[\bar{X}]$. We consider the problem of computing the global infimum of f when f is bounded below. For $\mathbf{A} \in \text{GL}_n(\mathbb{C})$, we denote by $f^{\mathbf{A}}$ the polynomial $f(\mathbf{A}\bar{X})$. Fix a number $M \in \mathbb{R}$ greater than $\inf_{x \in \mathbb{R}^n} f(x)$. We prove that there exists a Zariski-closed subset $\mathcal{A} \subsetneq \text{GL}_n(\mathbb{C})$ such that for all $\mathbf{A} \in \text{GL}_n(\mathbb{Q}) \setminus \mathcal{A}$, we have $f^{\mathbf{A}} \geq 0$ on $\mathbb{R}^n$ if and only if for all $\epsilon > 0$, there exist sums of squares of polynomials s and t in $\mathbb{R}[\bar{X}]$ and polynomials $\phi_i \in \mathbb{R}[\bar{X}]$ such that $f^{\mathbf{A}} + \epsilon = s + t(M - f^{\mathbf{A}}) + \sum_{1 \leq i \leq n-1} \phi_i \frac{\partial f^{\mathbf{A}}}{\partial X_i}$. Hence we can formulate the original optimization problems as semidefinite programs which can be solved efficiently in Matlab. Some numerical experiments are given. We also discuss how to exploit the sparsity of SDP problems to overcome the ill-conditionedness of SDP problems when the infimum is not attained.

Categories and Subject Descriptors

G.1.6 [Numerical Analysis]: Optimization; I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms: algebraic algorithms; F.2.2 [Analysis of Algorithms and Problem Complexity]: Non numerical algorithms and problems: geometrical problems and computation

General Terms

Theory, Algorithms

*Feng Guo and Lihong Zhi are supported by the Chinese National Natural Science Foundation under grant NSFC60821002/F02 and 10871194.

Feng Guo, Mohab Safey El Din and Lihong Zhi are supported by the EXACTA grant of the National Science Foundation of China (NSFC 60911130369) and the French National Research Agency (ANR-09-BLAN-0371-01).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

Keywords

Global optimization, polynomials, generalized critical values, sum of squares, semidefinite programming, moment matrix

1. INTRODUCTION

We consider the global optimization problem

$$f^* := \inf\{f(x) \mid x \in \mathbb{R}^n\} \in \mathbb{R} \cup \{-\infty\} \quad (1)$$

where $f \in \mathbb{R}[\bar{X}] := \mathbb{R}[X_1, \dots, X_n]$. The problem is equivalent to compute

$$f^* = \sup\{a \in \mathbb{R} \mid f - a \geq 0 \text{ on } \mathbb{R}^n\} \in \mathbb{R} \cup \{-\infty\}.$$

It is well known that this optimization problem is NP-hard even when $\deg(f) \geq 4$ and is even [13]. There are many approaches to approximate f^* . For example, we can get a lower bound by solving the sum of squares (SOS) problem:

$$\begin{aligned} f^{\text{SOS}} &= \sup\{a \in \mathbb{R} \mid f - a \text{ is a sum of squares in } \mathbb{R}[\bar{X}]\} \\ &\in \mathbb{R} \cup \{-\infty\}. \end{aligned}$$

The SOS problem can be solved efficiently by algorithms in GloptiPoly [4], SOSTOOLS [15], YALMIP [12], SeDuMi [22] and SparsePOP [24]. An overview about SOS and nonnegative polynomials is given in [17]. However, it is pointed out in [1] that for fixed degree $d \geq 4$, the volume of the set of sums of squares of polynomials in the set of nonnegative polynomials tends to 0 when the number of variable increases.

In recent years, a lot of work has been done in proving existence of SOS certificates which can be exploited for optimization, e.g., the “Big ball” method proposed by Lasserre [10] and “Gradient perturbation” method proposed by Jibetean and Laurent [6]. These two methods solve the problem by perturbing the coefficients of the input polynomials. However, small perturbations of coefficients might generate numerical instability and lead to SDPs which are hard to solve. The “Gradient variety” method by Nie, Demmel and Sturmfels [14] is an approach without perturbation. For a polynomial $f \in \mathbb{R}[\bar{X}]$, its *gradient variety* is defined as

$$V(\nabla f) := \{x \in \mathbb{C}^n \mid \nabla f(x) = 0\}$$

and its *gradient ideal* is the ideal generated by all partial derivatives of f :

$$\langle \nabla f \rangle := \left\langle \frac{\partial f}{\partial X_1}, \frac{\partial f}{\partial X_2}, \dots, \frac{\partial f}{\partial X_n} \right\rangle \subseteq \mathbb{R}[\bar{X}].$$

It is shown in [14] that if the polynomial $f \in \mathbb{R}[\bar{X}]$ is nonnegative on $V(\nabla f)$ and $\langle \nabla f \rangle$ is radical then f is an SOS modulo its gradient ideal. If the gradient ideal is not necessarily radical, the conclusion still holds for polynomials positive on their gradient variety. However, if f does not attain the infimum, the method outlined in [14] may provide a wrong answer. For example, consider $f := (1 - xy)^2 + y^2$. The infimum of f is $f^* = 0$, but $V(\nabla f) = \{(0, 0)\}$ and $f(0, 0) = 1$. This is due to the fact that any sequence (x_n, y_n) such that $f(x_n, y_n) \rightarrow 0$ when $n \rightarrow \infty$ satisfies $\|(x_n, y_n)\| \rightarrow \infty$ (here and throughout the paper we use the l^2 -norm). Roughly speaking the infimum is not reached at finite distance but “at infinity”. Such phenomena are related to the presence of *asymptotic* critical values, which is a notion introduced in [9].

Recently, there are some progress in dealing with these hard problems for which polynomials do not attain a minimum on $\mathbb{R}^n$. Let us outline Schweighofer’s approach [21]. We recall some notations firstly.

DEFINITION 1.1. *For any polynomial $f \in \mathbb{R}[\bar{X}]$ and subset $S \in \mathbb{R}^n$, the set $R_\infty(f, S)$ of asymptotic values of f on S consists of all $y \in \mathbb{R}$ for which there exists a sequence $(x_k)_{k \in \mathbb{N}}$ of points $x_k \in S$ such that $\lim_{k \rightarrow \infty} \|x_k\| = \infty$ and $\lim_{k \rightarrow \infty} f(x_k) = y$.*

DEFINITION 1.2. *The preordering generated by polynomials $g_1, g_2, \dots, g_m \in \mathbb{R}[\bar{X}]$ is denoted by $T(g_1, g_2, \dots, g_m)$:*

$$T(g_1, g_2, \dots, g_m) := \left\{ \sum_{\delta \in \{0,1\}^m} s_\delta g_1^{\delta_1} g_2^{\delta_2} \dots g_m^{\delta_m} \mid s_\delta \text{ is a sum of squares in } \mathbb{R}[\bar{X}] \right\}.$$

THEOREM 1.3. ([21, Theorem 9]). *Let $f, g_1, g_2, \dots, g_m \in \mathbb{R}[\bar{X}]$ and set*

$$S := \{x \in \mathbb{R}^n \mid g_1(x) \geq 0, g_2(x) \geq 0, \dots, g_m(x) \geq 0\}. \quad (2)$$

Suppose that

- (i) f is bounded on S ;
- (ii) $R_\infty(f, S)$ is a finite subset of $]0, +\infty[$;
- (iii) $f > 0$ on S ;

Then $f \in T(g_1, g_2, \dots, g_m)$.

The idea in [21] is to replace the real part $V(\nabla f) \cap \mathbb{R}^n$ of the gradient variety by several larger semialgebraic sets on which the partial derivatives do not necessarily vanish but get very small far away from the origin. For these sets two things must hold at the same time:

- There exist suitable SOS certificates for nonnegative polynomials on the set.
- The infimum of f on $\mathbb{R}^n$ and on the set coincide.

DEFINITION 1.4. *For a polynomial $f \in \mathbb{R}[\bar{X}]$, we call*

$$S(\nabla f) := \{x \in \mathbb{R}^n \mid 1 - \|\nabla f(x)\|^2 \|x\|^2 \geq 0\}$$

the principal gradient tentacle of f .

THEOREM 1.5. ([21, Theorem 25]) *Let $f \in \mathbb{R}[\bar{X}]$ be bounded below. Furthermore, suppose that f has only isolated singularities at infinity (which is always true in the case $n = 2$) or the principal gradient tentacle $S(\nabla f)$ is compact, then the following conditions are equivalent:*

- (i) $f \geq 0$ on $\mathbb{R}^n$;
- (ii) $f \geq 0$ on $S(\nabla f)$;
- (iii) *For every $\epsilon > 0$, there are sums of squares of polynomials s and t in $\mathbb{R}[\bar{X}]$, such that*

$$f + \epsilon = s + t \left(1 - \|\nabla f(\bar{X})\|^2 \|\bar{X}\|^2 \right).$$

For fixed $k \in \mathbb{N}$, let us define

$$f_k^* := \sup \{a \in \mathbb{R} \mid f - a = s + t (1 - \|\nabla f(x)\|^2 \|x\|^2)\}.$$

where s, t are sums of squares of polynomials and the degree of t is at most $2k$. If the assumptions in the above theorem are satisfied, then $\{f_k^*\}_{k \in \mathbb{N}}$ converges monotonically to f^* (see [21, Theorem 30]). The shortage of this method is that it is not clear that these technical assumptions are necessary or not. To avoid it, the author proposed a collection of higher gradient tentacles ([21, Definition 41]) defined by the polynomial inequalities

$$1 - \|\nabla f(x)\|^{2N} (1 + \|x\|)^{N+1} \geq 0, \quad N \in \mathbb{N}.$$

Then for sufficiently large N , for all $f \in \mathbb{R}[\bar{X}]$ bounded below we have an SOS representation theorem ([21, Theorem 46]). However, the corresponding SDP relaxations get very large for large N and one has to deal for each N with a sequence of SDPs. To avoid this disadvantage, another approach using *truncated tangency variety* is proposed in [3]. Their results are mainly based on Theorem 1.3. For nonconstant polynomial function $f \in \mathbb{R}[\bar{X}]$, they define

$$g_{ij}(\bar{X}) := X_j \frac{\partial f}{\partial X_i} - X_i \frac{\partial f}{\partial X_j}, \quad 1 \leq i < j \leq n.$$

For a fixed real number $M \in f(\mathbb{R}^n)$, the *truncated tangency variety* of f is defined to be

$$\Gamma_M(f) := \{x \in \mathbb{R}^n \mid M - f(x) \geq 0, g_{i,j}(x) = 0, 1 \leq i, j \leq n\}.$$

Then based on Theorem 1.3, the following result is proved.

THEOREM 1.6. [3, Theorem 3.1] *Let $f \in \mathbb{R}[\bar{X}]$ and M be a fixed real number. Then the following conditions are equivalent:*

- (i) $f \geq 0$ on $\mathbb{R}^n$;
- (ii) $f \geq 0$ on $\Gamma_M(f)$;
- (iii) *For every $\epsilon > 0$, there are sums of squares of polynomials s and t in $\mathbb{R}[\bar{X}]$ and polynomials $\phi_{ij} \in \mathbb{R}[\bar{X}]$, $1 \leq i < j \leq n$, such that*

$$f + \epsilon = s + t(M - f) + \sum_{1 \leq i < j \leq n} \phi_{ij} g_{ij}.$$

Fix $k \in \mathbb{N}$ and let

$$f_k^* := \sup \left\{ a \in \mathbb{R} \mid f - a = s + t(M - f) + \sum_{1 \leq i < j \leq n} \phi_{ij} g_{ij} \right\}.$$

where s, t, ϕ_{ij} are polynomials of degree at most $2k$ and s, t are sums of squares of polynomials in $\mathbb{R}[\bar{X}]$, then the sequence $\{f_k^*\}_{k \in \mathbb{N}}$ converges monotonically increasing to f^* ([3, Theorem 3.2]). This approach does not require the assumptions of [21, Theorem 25]. However, the number of

equality constraints in $\Gamma_M(f)$ is $\frac{n(n-1)}{2}$ which is very large as n increases.

In this paper, based on Theorem 1.3 and the computation of *generalized critical values* of a polynomial mapping in [18, 19], we present a method to solve optimization problem (1) without requiring f attains the infimum on $\mathbb{R}^n$. Our method does not require assumptions as in [21] and use the simpler variety which only contains $n - 1$ equality constraints.

Although approaches in [21] and [3] can handle polynomials which do not attain a minimum on $\mathbb{R}^n$, numerical problems occur when one solves the SDPs obtained from SOS relaxations, see [3, 6, 21]. The numerical problems are mainly caused by the unboundness of the moments. It happens often when one deals with this kind of polynomial optimization problem using SOS relaxations without exploiting the sparsity structure. We propose some strategies to avoid ill-conditionedness of moment matrices.

The paper is organized as follows. In section 2 we present some notations and preliminaries used in our method. The main result and its proof are given in section 3. In section 4, some numerical experiments are given. In section 5, we focus on two polynomials which do not attain the infimum and try to solve the numerical problems. We draw some conclusions in section 6.

2. PRELIMINARIES AND NOTATIONS

DEFINITION 2.1. [9] A complex (resp. real) number $c \in \mathbb{C}$ (resp. $c \in \mathbb{R}$) is a critical value of the mapping $\tilde{f}_{\mathbb{C}} : x \in \mathbb{C}^n \rightarrow f(x)$ (resp. $\tilde{f}_{\mathbb{R}} : x \in \mathbb{R}^n \rightarrow f(x)$) if and only if there exists $z \in \mathbb{C}^n$ (resp. $z \in \mathbb{R}^n$) such that $f(z) = c$ and $\frac{\partial f}{\partial X_1} = \dots = \frac{\partial f}{\partial X_n} = 0$.

A complex (resp. real) number $c \in \mathbb{C}$ (resp. $c \in \mathbb{R}$) is an asymptotic critical value of the mapping $\tilde{f}_{\mathbb{C}}$ (resp. $\tilde{f}_{\mathbb{R}}$) if there exists a sequence of points $(z_l)_{l \in \mathbb{N}} \subset \mathbb{C}^n$ (resp. $(z_l)_{l \in \mathbb{N}} \subset \mathbb{R}^n$) such that:

- (i) $f(z_l)$ tends to c when l tends to ∞ .
- (ii) $\|z_l\|$ tends to $+\infty$ when l tends to ∞ .
- (iii) $\|X_i(z_l)\| \cdot \|\frac{\partial f}{\partial X_j}(z_l)\|$ tends to 0 when l tends to ∞ for all $(i, j) \in \{1, \dots, n\}^2$.

We denote by $K_0(f)$ the set of critical values of f , by $K_\infty(f)$ the set of asymptotic critical values of f , and by $K(f)$ the set of generalized critical values which is the union of $K_0(f)$ and $K_\infty(f)$.

DEFINITION 2.2. A map $\phi : V \rightarrow W$ of topological spaces is said to be proper at $w \in W$ if there exists a neighborhood B of w such that $\phi^{-1}(\overline{B})$ is compact (where $\overline{B}$ denotes the closure of B).

Recall that for $\mathbf{A} \in \text{GL}_n(\mathbb{C})$ and $f \in \mathbb{R}[\bar{X}]$, $f^{\mathbf{A}}$ the polynomial $f(\mathbf{A} \bar{X})$.

LEMMA 2.3. ([18], Lemma 1) For all $\mathbf{A} \in \text{GL}_n(\mathbb{Q})$, we have $K_0(f) = K_0(f^{\mathbf{A}})$ and $K_\infty(f) = K_\infty(f^{\mathbf{A}})$.

THEOREM 2.4. ([18, Theorem 3.6]) There exists a Zariski-closed subset $\mathcal{A} \subsetneq \text{GL}_n(\mathbb{C})$ such that for all $\mathbf{A} \in \text{GL}_n(\mathbb{Q}) \setminus \mathcal{A}$, the set of real asymptotic critical values of $x \rightarrow f(x)$ is contained in the set of non-properness of the projection on T restricted to the Zariski-closure of the semi-algebraic set defined by $f^{\mathbf{A}} - T = \frac{\partial f^{\mathbf{A}}}{\partial X_1} = \dots = \frac{\partial f^{\mathbf{A}}}{\partial X_{n-1}} = 0$, $\frac{\partial f^{\mathbf{A}}}{\partial X_n} \neq 0$.

In [18], the above theorem is stated in the complex case and proved for this case. It relies on properness properties of some critical loci. Since these properness properties can be transferred to the real part of these critical loci, its proof can be transposed mutatis mutandis to the real case.

REMARK 2.5. ([19], Remark 1) Note also that the curve defined as the Zariski-closure of the complex solution set of $f^{\mathbf{A}} - T = \frac{\partial f^{\mathbf{A}}}{\partial X_1} = \dots = \frac{\partial f^{\mathbf{A}}}{\partial X_{n-1}} = 0$, $\frac{\partial f^{\mathbf{A}}}{\partial X_n} \neq 0$ has a degree bounded by $(d-1)^{(n-1)}$, where d is the total degree of f . Thus the set of the non-properness of the projection on T restricted to this curve has a degree bounded by $(d-1)^{(n-1)}$.

REMARK 2.6. In [18], a criterion for choosing $\mathbf{A}$ is given. It is sufficient that the restriction of the projection $(x_1, \dots, x_n, t) \rightarrow (x_{n-i+2}, \dots, x_n, t)$ to the Zariski-closure of the constructible set $f^{\mathbf{A}} - T = \frac{\partial f^{\mathbf{A}}}{\partial X_1} = \dots = \frac{\partial f^{\mathbf{A}}}{\partial X_{n-i}} = 0$, $\frac{\partial f^{\mathbf{A}}}{\partial X_{n-i+1}} \neq 0$ is proper. An algorithm, based on Gröbner bases or triangular sets computations, that computes sets of non-properness is given in [20].

THEOREM 2.7. ([19, Theorem 5]) Let $f \in \mathbb{R}[X_1, \dots, X_n]$ and $\varepsilon = \{e_1, \dots, e_l\}$ (with $e_1 < \dots < e_l$) be the set of real generalized critical values of the mapping $x \in \mathbb{R}^n \rightarrow f(x)$. Then $\inf_{x \in \mathbb{R}^n} f(x) > -\infty$ if and only if there exists $1 \leq i_0 \leq l$ such that $\inf_{x \in \mathbb{R}^n} f(x) = e_{i_0}$.

REMARK 2.8. Combined with Lemma 2.3, the above theorem leads to $f^* = \inf_{x \in \mathbb{R}^n} f^{\mathbf{A}}(x)$.

3. MAIN RESULTS

For $\mathbf{A} \in \text{GL}_n(\mathbb{Q})$, we denote by $W_1^{\mathbf{A}}$ the constructible set defined by

$$\left\{ \frac{\partial f^{\mathbf{A}}}{\partial X_1} = \dots = \frac{\partial f^{\mathbf{A}}}{\partial X_{n-1}} = 0, \frac{\partial f^{\mathbf{A}}}{\partial X_n} \neq 0 \right\},$$

and by $W_0^{\mathbf{A}}$ the algebraic variety defined by

$$\left\{ \frac{\partial f^{\mathbf{A}}}{\partial X_1} = \dots = \frac{\partial f^{\mathbf{A}}}{\partial X_{n-1}} = \frac{\partial f^{\mathbf{A}}}{\partial X_n} = 0 \right\}.$$

We set $W^{\mathbf{A}} := W_1^{\mathbf{A}} \cup W_0^{\mathbf{A}}$, which is the algebraic variety defined by

$$\left\{ \frac{\partial f^{\mathbf{A}}}{\partial X_1} = \dots = \frac{\partial f^{\mathbf{A}}}{\partial X_{n-1}} = 0 \right\}.$$

LEMMA 3.1. If $\inf_{x \in \mathbb{R}^n} f(x) > -\infty$, then there exists a Zariski-closed subset $\mathcal{A} \subsetneq \text{GL}_n(\mathbb{C})$ such that for all $\mathbf{A} \in \text{GL}_n(\mathbb{Q}) \setminus \mathcal{A}$,

$$f^* = \inf\{f(x) \mid x \in \mathbb{R}^n\} = \inf\{f^{\mathbf{A}}(x) \mid x \in W^{\mathbf{A}} \cap \mathbb{R}^n\}.$$

Moreover $R_\infty(f^{\mathbf{A}}, W^{\mathbf{A}})$ is a finite set.

Proof. We start by proving that $f^* = \inf\{f^{\mathbf{A}}(x) \mid x \in W^{\mathbf{A}}\}$. Remark first that $f^* \leq \inf\{f^{\mathbf{A}}(x) \mid x \in W^{\mathbf{A}} \cap \mathbb{R}^n\}$.

- Suppose first that the infimum f^* is reached over $\mathbb{R}^n$. Then, it is reached at a critical point $x \in W_0^{\mathbf{A}}$. Since $W_0^{\mathbf{A}} \subset W^{\mathbf{A}}$, $f^* = \inf\{f^{\mathbf{A}}(x) \mid x \in W^{\mathbf{A}} \cap \mathbb{R}^n\}$.
- Suppose now that the infimum f^* is not reached over $\mathbb{R}^n$. Then, by Theorems 2.4 and 2.7, f^* belongs to the

set of non-properness of the restriction of the projection $(x, t) \rightarrow t$ to the Zariski-closure of the set defined by

$$f^{\mathbf{A}} - T = \frac{\partial f}{\partial X_1} = \dots = \frac{\partial f}{\partial X_{n-1}} = 0, \quad \frac{\partial f}{\partial X_n} \neq 0.$$

This implies that for all $\varepsilon > 0$, there exists $(x, t) \in \mathbb{R}^n \times \mathbb{R}$ such that $x \in W_1^{\mathbf{A}} \cap \mathbb{R}^n$ and $f^* \leq t \leq f^* + \varepsilon$. This implies that $f^* \geq \inf\{f^{\mathbf{A}}(x) \mid x \in W^{\mathbf{A}} \cap \mathbb{R}^n\}$. We conclude that $f^* = \inf\{f^{\mathbf{A}}(x) \mid x \in W^{\mathbf{A}} \cap \mathbb{R}^n\}$ since we previously proved that $f^* \leq \inf\{f^{\mathbf{A}}(x) \mid x \in W^{\mathbf{A}} \cap \mathbb{R}^n\}$.

We prove now that $R_{\infty}(f^{\mathbf{A}}, W^{\mathbf{A}})$ is finite. Remark that $R_{\infty}(f^{\mathbf{A}}, W^{\mathbf{A}}) = R_{\infty}(f^{\mathbf{A}}, W_0^{\mathbf{A}}) \cup R_{\infty}(f^{\mathbf{A}}, W_1^{\mathbf{A}})$. The set

$$R_{\infty}(f^{\mathbf{A}}, W_0^{\mathbf{A}}) \subset \{f^{\mathbf{A}} \mid x \in W_0^{\mathbf{A}}\} = K_0(f^{\mathbf{A}})$$

is finite. Moreover, by Definition 1.1 and 2.2, $R_{\infty}(f^{\mathbf{A}}, W_1^{\mathbf{A}})$ is a subset of the non-properness set of the mapping $\tilde{f}$ restricted to $W_1^{\mathbf{A}}$, which by Remark 2.5 is a finite set. Hence $R_{\infty}(f^{\mathbf{A}}, W^{\mathbf{A}})$ is a finite set. $\square$

Fix a real number $M \in f(\mathbb{R}^n)$ and for all $\mathbf{A} \in GL_n(\mathbb{Q})$, consider the following semi-algebraic set

$$W_M^{\mathbf{A}} = \left\{x \in \mathbb{R}^n \mid M - f^{\mathbf{A}}(x) \geq 0, \frac{\partial f^{\mathbf{A}}}{\partial X_i} = 0, 1 \leq i \leq n-1\right\}.$$

LEMMA 3.2. *There exists a Zariski-closed subset $\mathcal{A} \subsetneq GL_n(\mathbb{C})$ such that for all $\mathbf{A} \in GL_n(\mathbb{Q}) \setminus \mathcal{A}$, if*

$$\inf\{f^{\mathbf{A}}(x) \mid x \in W_M^{\mathbf{A}}\} > 0,$$

then $f^{\mathbf{A}}$ can be written as a sum

$$f^{\mathbf{A}} = s + t \left(M - f^{\mathbf{A}} \right) + \sum_{1 \leq i \leq n-1} \phi_i \frac{\partial f^{\mathbf{A}}}{\partial X_i}, \quad (3)$$

where $\phi_i \in \mathbb{R}[\bar{X}]$ for $1 \leq i \leq n-1$, and s, t are sums of squares in $\mathbb{R}[\bar{X}]$.

Proof. By Lemma 3.1, $f^{\mathbf{A}}$ is bounded, positive on $W_M^{\mathbf{A}}$. By Lemma 3.1, $R_{\infty}(f^{\mathbf{A}}, W_M^{\mathbf{A}})$ is a finite set. Then, Theorem 1.3 implies that $f^{\mathbf{A}}$ can be written as a sum (3). $\square$

THEOREM 3.3. *Let $f \in \mathbb{R}[\bar{X}]$ be bounded below, and $M \in f(\mathbb{R}^n)$. There exists a Zariski-closed subset $\mathcal{A} \subsetneq GL_n(\mathbb{C})$ such that for all $\mathbf{A} \in GL_n(\mathbb{Q}) \setminus \mathcal{A}$, the following conditions are equivalent:*

- (i) $f^{\mathbf{A}} \geq 0$ on $\mathbb{R}^n$;
- (ii) $f^{\mathbf{A}} \geq 0$ on $W_M^{\mathbf{A}}$;
- (iii) For every $\epsilon > 0$, there are sums of squares of polynomials s and t in $\mathbb{R}[\bar{X}]$ and polynomials $\phi_i \in \mathbb{R}[\bar{X}]$, $1 \leq i \leq n-1$, such that

$$f^{\mathbf{A}} + \epsilon = s + t \left(M - f^{\mathbf{A}} \right) + \sum_{1 \leq i \leq n-1} \phi_i \frac{\partial f^{\mathbf{A}}}{\partial X_i}.$$

Proof. By Lemma 3.2 and Theorem 1.3. $\square$

DEFINITION 3.4. *For all polynomials $f \in \mathbb{R}[\bar{X}]$, denote by d the total degree of f . Then for all $k \in \mathbb{N}$, we define*

$f_k^* \in \mathbb{R} \cup \{\pm\infty\}$ as the supremum over all $a \in \mathbb{R}$ such that $f - a$ can be written as a sum

$$f - a = s + t(M - f) + \sum_{1 \leq i \leq n-1} \phi_i \frac{\partial f}{\partial X_i} \quad (4)$$

where $t, \phi_i, 1 \leq i \leq n-1$ are polynomials of degree at most $2k$, for $k \in \mathbb{N}$ and s, t are sums of squares of polynomials in $\mathbb{R}[\bar{X}]$.

THEOREM 3.5. *Let $f \in \mathbb{R}[\bar{X}]$ be bounded below. Then there exists a Zariski-closed subset $\mathcal{A} \subsetneq GL_n(\mathbb{C})$ such that for all $\mathbf{A} \in GL_n(\mathbb{Q}) \setminus \mathcal{A}$, the sequence $\{f_k^*\}, k \in \mathbb{N}$ converges monotonically increasing to the infimum $(f^{\mathbf{A}})^*$ which equals to f^* by Lemma 2.8.*

4. NUMERICAL RESULTS

Examples below are cited from [3, 6, 10, 14, 21]. We use Matlab package SOSTOOLS [15] to compute optimal values f_k^* by relaxations of order k over

$$W_M^{\mathbf{A}} = \left\{x \in \mathbb{R}^n \mid M - f^{\mathbf{A}}(x) \geq 0, \frac{\partial f^{\mathbf{A}}}{\partial X_i} = 0, 1 \leq i \leq n-1\right\}.$$

In the following test, we set $\mathbf{A} := I_{n \times n}$ be an identity matrix, and without loss of generality, we let $M := f^{\mathbf{A}}(0) = f(0)$. The set $W_M^{\mathbf{A}}$ is very simple and the results we get are very similar to or better than the given results in literatures [3, 6, 10, 14, 21].

EXAMPLE 4.1. *Let us consider the polynomial*

$$f(x, y) := (xy - 1)^2 + (x - 1)^2.$$

Obviously, $f^* = f^{sos} = 0$ which can be reached at $(1, 1)$. The computed optimal values are $f_0^* \approx 0.34839 \cdot 10^{-8}$, $f_1^* \approx 0.16766 \cdot 10^{-8}$ and $f_2^* \approx 0.29125 \cdot 10^{-8}$.

EXAMPLE 4.2. *Let us consider the Motzkin polynomial*

$$f(x, y) := x^2 y^4 + x^4 y^2 - 3x^2 y^2 + 1.$$

It is well known that $f^* = 0$ but $f^{sos} = -\infty$. The computed optimal values are $f_0^* \approx -6138.2$, $f_1^* \approx -52508$, $f_2^* \approx 0.15077 \cdot 10^{-8}$ and $f_3^* \approx 0.36591 \cdot 10^{-8}$.

EXAMPLE 4.3. *Let us consider the Berg polynomial*

$$f(x, y) := x^2 y^2 (x^2 + y^2 - 1).$$

We know that $f^* = -1/27 \approx -0.037037037$. But $f^{sos} = -\infty$. Our computed optimal values are $f_0^* \approx -563.01$, $f_1^* \approx -0.056591$, $f_2^* \approx -0.037037$ and $f_3^* \approx -0.037037$.

EXAMPLE 4.4. *Let*

$$f(x, y) := (x^2 + 1)^2 + (y^2 + 1)^2 - 2(x + y + 1)^2.$$

Since f is a bivariate polynomial of degree 4, $f - f^*$ must be a sum of squares. By computation, we obtain f_0^*, f_1^*, f_2^* all approximately equal to -11.458 .

EXAMPLE 4.5. *Consider the polynomial of three variables:*

$$f(x, y, z) := (x + x^2 y + x^4 y z)^2.$$

As mentioned in [21], this polynomial has non-isolated singularities at infinity. It is clear that $f^* = 0$. Our computed optimal values are: $f_0^* \approx -0.36282 \cdot 10^{-8}$, $f_1^* \approx -0.31482 \cdot 10^{-7}$, $f_2^* \approx -0.1043 \cdot 10^{-7}$ and $f_3^* \approx -0.58405 \cdot 10^{-8}$.

EXAMPLE 4.6. Let us consider the homogenous Motzkin polynomial of three real variables:

$$f(x, y, z) := x^2 y^2 (x^2 + y^2 - 3z^2) + z^6.$$

It is known that $f^* = 0$ but $f^{sos} = -\infty$. By computation, we get optimal values: $f_0^* \approx -0.27651$, $f_1^* \approx -0.13287 \cdot 10^{-2}$, $f_2^* \approx -0.19772 \cdot 10^{-3}$, $f_3^* \approx -0.95431 \cdot 10^{-4}$, $f_4^* \approx -0.60821 \cdot 10^{-4}$, $f_5^* \approx -0.32235 \cdot 10^{-4}$ and $f_6^* \approx -0.2625 \cdot 10^{-4}$.

EXAMPLE 4.7. Consider the polynomial from [11]

$$f := \sum_{i=1}^5 \prod_{j \neq i} (X_i - X_j) \in \mathbb{R}[X_1, X_2, X_3, X_4, X_5].$$

It is shown in [11] that $f^* = 0$ but $f^{sos} = -\infty$. In [21], the results computed using gradient tentacles are $f_0^* \approx -0.2367$, $f_1^* \approx -0.0999$ and $f_2^* \approx -0.0224$. Using truncated tangency variety in [3], we get $f_0^* \approx -1.9213$, $f_1^* \approx -0.077951$ and $f_2^* \approx -0.015913$. The optimal values we computed are better: $f_0^* \approx -4.4532$, $f_1^* \approx -0.43708 \cdot 10^{-7}$, $f_2^* \approx -0.21811 \cdot 10^{-6}$. The number of equality constraints in [3] is 10 while we only add 4 equality constraints.

EXAMPLE 4.8. Let us consider the following example of Robinson [17].

$$R(x, y, 1) := x^6 + y^6 + 1 - (x^4 y^2 + x^2 y^4 + x^4 + x^2 + y^4 + y^2) + 3x^2 y^2.$$

It is proved that $f^* = 0$ but $f^{sos} = -\infty$. Our computed lower bounds are: $f_0^* \approx -0.9334$, $f_1^* \approx -0.23408$, $f_2^* \approx -0.22162 \times 10^{-2}$ and $f_3^* \approx 0.88897 \times 10^{-9}$.

5. UNATTAINABLE INFIMUM VALUE

EXAMPLE 5.1. Consider the polynomial

$$f(x, y) := (1 - xy)^2 + y^2.$$

The polynomial f does not attain its infimum $f^* = 0$ on $\mathbb{R}^2$. Since f is a sum of squares, we have $f^{sos} = 0$ and therefore $f_k^* = 0$ for all $k \in \mathbb{N}$. However, as shown in [3, 6, 21], there are always numerical problems. For example, the results given in [3] are $f^{sos} \approx 1.5142 \cdot 10^{-12}$, $f_0^* \approx -0.12641 \cdot 10^{-3}$, $f_1^* \approx 0.12732 \cdot 10^{-1}$, $f_2^* \approx 0.49626 \cdot 10^{-1}$.

For polynomials which do not attain their infimum values, we investigate the numerical problem involved in solving the SOS relaxation:

$$\sup \left\{ a \mid f - a = m_d(\bar{X})^T \cdot W \cdot m_d(\bar{X}), W \succeq 0, W^T = W \right\}, \quad (5)$$

where $m_d(\bar{X})$ is a vector of monomials of degree less than or equal to d , W is also called the Gram matrix.

SDPTools is a package for solving SDPs in Maple [2]. It includes an SDP solver which implements the classical primal-dual potential reduction algorithm [23]. This algorithm requires initial strictly feasible primal and dual points. Usually, it is difficult to find a strictly feasible point for (5). According to the Big-M method, after introducing two big positive numbers M_1 and M_2 , we convert (5) to the following

# iter.	prec.	gap	lower bound r	M_1	M_2
50	75	.74021e-17	.46519e-1	10^3	10^3
50	75	.12299e-11	.47335e-2	10^3	10^5
50	75	.68693e-12	.47335e-2	10^5	10^5
50	75	.38601e-10	.47424e-3	10^3	10^7
70	75	.76145e-18	.47424e-3	10^7	10^7
50	75	.43114e-10	.47433e-4	10^3	10^9
70	75	.33233e-12	.47433e-4	10^9	10^9
75	90	.86189e-10	.47426e-5	10^3	10^{11}

Table 1: Lower bounds with $m_d(\bar{X}) = [1, x, y, x^2, xy, y^2]^T$

form:

$$\left. \begin{aligned} & \sup_{\hat{r} \in \mathbb{R}, \hat{W}} \hat{r} - M_2 z \\ & \text{s.t. } f(\bar{X}) - \hat{r} + z(m_d(\bar{X})^T \cdot m_d(\bar{X})) \\ & \quad = m_d(\bar{X})^T \cdot \hat{W} \cdot m_d(\bar{X}), \\ & \quad \hat{W} \succeq 0, \quad \hat{W}^T = \hat{W}, \quad z \geq 0, \\ & \quad \text{Tr}(\hat{W}) \leq M_1. \end{aligned} \right\} \quad (6)$$

The dual form of (6) is

$$\left. \begin{aligned} & \inf_{y_\alpha, t \in \mathbb{R}} \sum_{\alpha} f_{\alpha} y_{\alpha} + M_1 t \\ & \text{s.t. } \text{Moment}_d(y) + tI \succeq 0, \quad t \geq 0 \\ & \quad \text{Tr}(\text{Moment}_d(y)) \leq M_2. \end{aligned} \right\} \quad (7)$$

Assuming the primal and dual problems are both bounded, suppose M_1 and M_2 are chosen larger than the upper bounds on the traces of the Gram matrix and the moment matrix respectively, then this entails no loss of generality. In practice, these upper bounds are not known, and we can only guess some appropriate values for M_1, M_2 from the given polynomials. If we can not get the right results, we will increase M_1, M_2 and solve the SDPs again.

In Table 1, we choose $m_d(\bar{X}) := [1, x, y, x^2, xy, y^2]^T$ and solve (6) and (7) for different M_1 and M_2 .

The first column is the number of iterations and the second column is the number of digits we used in Maple. The third column is the gap of the primal and dual SDPs at the solutions. It is clear that the corresponding SDPs can be solved quite accurately with enough number of iterations. However, the lower bounds we get are not so good. If we choose larger M_2 , the lower bound becomes better. As mentioned earlier, the number M_2 is chosen as the upper bound on the trace of the moment matrix at the optimizers. So it implies that the trace of the corresponding moment matrix may be unbounded. Let us consider the primal and dual SDPs obtained from SOS relaxation of (1):

$$\mathbb{P} \mapsto \left\{ \begin{aligned} & \inf_{y_{\alpha} \in \mathbb{R}} \sum_{\alpha} f_{\alpha} y_{\alpha} \\ & \text{s.t. } \text{Moment}_d(y) \succeq 0. \end{aligned} \right. \quad (8)$$

$$\mathbb{P}^* \mapsto \left\{ \begin{aligned} & \sup_{r \in \mathbb{R}} r \\ & \text{s.t. } f(\bar{X}) - r = m_d(\bar{X})^T \cdot W \cdot m_d(\bar{X}), \\ & \quad W \succeq 0, \quad W^T = W. \end{aligned} \right. \quad (9)$$

For Example 5.1, f is a sum of squares, so $\mathbb{P}^*$ has a feasible solution. By proposition 3.1 in [10], $\mathbb{P}^*$ is solvable and $\inf \mathbb{P} =$

$\max \mathbb{P}^* = 0$. We show that for $m_d(\bar{X}) = [1, x, y, x^2, xy, y^2]^T$, $\mathbb{P}$ does not attain the minimum. To the contrast, if y^* is a minimizer of the SDP problem $\mathbb{P}$, then we have

$$1 - 2y_{1,1} + y_{2,2} + y_{0,2} = 0, \quad (10)$$

and

$$\text{Moment}_2(y) = \begin{bmatrix} 1 & y_{1,0} & y_{0,1} & y_{2,0} & y_{1,1} & y_{0,2} \\ y_{1,0} & y_{2,0} & y_{1,1} & y_{3,0} & y_{2,1} & y_{1,2} \\ y_{0,1} & y_{1,1} & y_{0,2} & y_{2,1} & y_{1,2} & y_{0,3} \\ y_{2,0} & y_{3,0} & y_{2,1} & y_{4,0} & y_{3,1} & y_{2,2} \\ y_{1,1} & y_{2,1} & y_{1,2} & y_{3,1} & y_{2,2} & y_{1,3} \\ y_{0,2} & y_{1,2} & y_{0,3} & y_{2,2} & y_{1,3} & y_{0,4} \end{bmatrix} \succeq 0.$$

Since $\text{Moment}_2(y)$ is a positive semidefinite matrix, we have $y_{0,2} \geq 0$ and $|2y_{1,1}| \leq (1 + y_{2,2})$. Combining with (10), we must have $y_{0,2} = 0$ and

$$2y_{1,1} = 1 + y_{2,2}. \quad (11)$$

Because $\text{Moment}_2(y)$ is positive semidefinite, from $y_{0,2} = 0$, we can derive $y_{1,1} = 0$. Therefore, by (11), we have $y_{2,2} = -1$. It is a contradiction.

Let us show that the dual problem of (9) is not bounded if we choose $m_d(\bar{X}) = [1, x, y, x^2, xy, y^2]^T$. The infimum of $f(x, y)$ can only be reached at “infinity”: $p^* = (x^*, y^*) \in \{\mathbb{R} \cup \pm\infty\}^2$. The vector

$$[x^*, y^*, x^{*2}, x^*y^*, y^{*2}, x^{*3}, x^{*2}y^*, x^*y^{*2}, y^{*3}, x^{*4}, x^{*3}y^*, x^{*2}y^{*2}, x^*y^{*3}, y^{*4}]$$

is a minimizer of (8) at “infinity”. Since $x^*y^* \rightarrow 1$ and $y^* \rightarrow 0$, when $\|(x^*, y^*)\|$ goes to ∞ , any moment $y_{i,j}$ with $i > j$ tends to ∞ . So the trace of the moment matrix tends to ∞ .

If we increase the bound M_2 , we can get better results as shown in Table 1. For example, by setting $M_1 = 10^3, M_2 = 10^{11}$, we get $f^* = 0.4743306 \times 10^{-5}$. However this method converges very slowly at the beginning and needs large amount of computations.

THEOREM 5.2. [16] For a polynomial $p(x) = \sum_{\alpha} p_{\alpha} x^{\alpha}$, we define $C(p)$ as the convex hull of $\text{sup}(p) = \{\alpha | p_{\alpha} \neq 0\}$, then we have $C(p^2) = 2C(p)$; for any positive semidefinite polynomials f and g , $C(f) \subseteq C(f + g)$; if $f = \sum_j g_j^2$ then $C(g_j) \subseteq \frac{1}{2}C(f)$.

For the polynomial f in Example 5.1, $C(f)$ is the convex hull of the points $(0, 0), (1, 1), (0, 2), (2, 2)$; see Figure 1. According to Theorem 5.2, the SOS decomposition of f contains only monomials whose supports are $(0, 0), (0, 1), (1, 1)$. Hence, if we choose a sparse monomial vector $m_d(\bar{X}) = [1, y, xy]^T$, for $M_1 = 1000$ and $M_2 = 1000$, from Table 2, we can see a very accurate optimal value is obtained. This is due to the fact that the trace of the moment matrix at the optimizer (x^*, y^*) now is $1 + y^{*2} + x^{*2}y^{*2}$, which is bounded when x^*y^* goes to 1 and y^* goes to 0. That is the main reason that we get very different results in Table 1 and 2. We can also verify the above results by using `solvesos` in YALMIP[12]; see Table 3.

In the following, in order to remove the monomials which cause the ill-conditionedness of the moment matrix, we also try to exploit the sparsity structure when we compute optimal values f_k^* by SOS relaxations of order k over W_M^A .

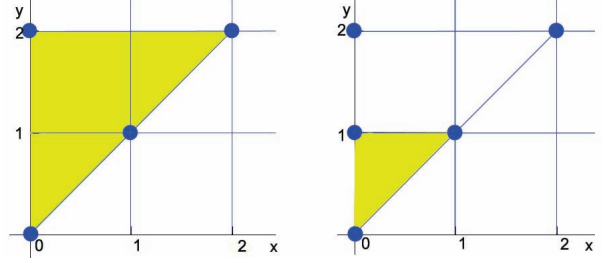


Figure 1: Newton polytope for the polynomial f (left), and the possible monomials in its SOS decomposition (right).

# iter.	prec.	gap	lower bound r	M_1	M_2
50	75	.97565e-27	-.38456e-28	10^3	10^3

Table 2: The lower bounds using $m_d(\bar{X}) = [1, y, xy]^T$

Let $A = I_{2 \times 2}$, $m_{d_1}(\bar{X}) = m_{d_2}(\bar{X}) := [1, x, y, x^2, xy, y^2]$, and symmetric semidefinite positive matrices W, V satisfying

$$\begin{aligned} f + \epsilon &= m_{d_1}(\bar{X})^T \cdot W \cdot m_{d_1}(\bar{X}) \\ &+ m_{d_2}(\bar{X})^T \cdot V \cdot m_{d_2}(\bar{X}) \cdot (M - f) + \phi \frac{\partial f}{\partial x}. \end{aligned}$$

Hence

$$\begin{aligned} f + \epsilon &\equiv m_{d_1}(\bar{X})^T \cdot W \cdot m_{d_1}(\bar{X}) \\ &+ m_{d_2}(\bar{X})^T \cdot V \cdot m_{d_2}(\bar{X}) \cdot (M - f) \bmod J, \end{aligned} \quad (12)$$

where $J = \langle \frac{\partial f}{\partial x} \rangle$.

For simplicity, we choose $M = 5$. If we do not exploit the sparsity structure, the associated moment matrix is a diagonal matrix $\begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix}$, where

$$P = \begin{bmatrix} y_{0,0} & y_{1,0} & y_{0,1} & y_{2,0} & y_{1,1} & y_{0,2} \\ y_{1,0} & y_{2,0} & y_{1,1} & y_{3,0} & y_{2,1} & y_{1,2} \\ y_{0,1} & y_{1,1} & y_{0,2} & y_{2,1} & y_{1,2} & y_{0,3} \\ y_{2,0} & y_{3,0} & y_{2,1} & y_{4,0} & y_{3,1} & y_{2,2} \\ y_{1,1} & y_{2,1} & y_{1,2} & y_{3,1} & y_{2,2} & y_{1,3} \\ y_{0,2} & y_{1,2} & y_{0,3} & y_{2,2} & y_{1,3} & y_{0,4} \end{bmatrix}$$

and $Q =$

$$\begin{bmatrix} 4y_{0,0} + y_{1,1} - y_{0,2} & 4y_{1,0} - y_{0,1} + y_{2,1} & 5y_{0,1} - y_{0,3} \\ 4y_{1,0} - y_{0,1} + y_{2,1} & y_{3,1} - y_{1,1} + 4y_{2,0} & 5y_{1,1} - y_{0,2} \\ 5y_{0,1} - y_{0,3} & 5y_{1,1} - y_{0,2} & 5y_{0,2} - y_{0,4} \\ y_{3,1} - y_{1,1} + 4y_{2,0} & -y_{2,1} + 4y_{3,0} + y_{4,1} & 5y_{2,1} - y_{0,1} \\ 5y_{1,1} - y_{0,2} & 5y_{2,1} - y_{0,1} & 5y_{0,1} - y_{0,3} \\ 5y_{0,2} - y_{0,4} & 5y_{0,1} - y_{0,3} & 5y_{0,3} - y_{0,5} \\ y_{3,1} - y_{1,1} + 4y_{2,0} & 5y_{1,1} - y_{0,2} & 5y_{0,2} - y_{0,4} \\ -y_{2,1} + 4y_{3,0} + y_{4,1} & 5y_{2,1} - y_{0,1} & 5y_{0,1} - y_{0,3} \\ 5y_{2,1} - y_{0,1} & 5y_{0,1} - y_{0,3} & 5y_{0,3} - y_{0,5} \\ -y_{3,1} + 4y_{4,0} + y_{5,1} & 5y_{3,1} - y_{1,1} & 5y_{1,1} - y_{0,2} \\ 5y_{3,1} - y_{1,1} & 5y_{1,1} - y_{0,2} & 5y_{0,2} - y_{0,4} \\ 5y_{1,1} - y_{0,2} & 5y_{0,2} - y_{0,4} & 5y_{0,4} - y_{0,6} \end{bmatrix}$$

$m_d(\bar{X})$	lower bounds r
$[1, y, xy]^T$	.14853e-11
$[1, x, y, xy]^T$	.414452e-4
$[1, x, y, x^2, xy, y^2]^T$	.15952e-2

Table 3: The lower bounds using `solvesos` in Matlab

We can see that the moment matrix has lots of terms $y_{i,j}$ for $i > j$ which tend to infinity when we get close to the optimizer.

In the following we will try to remove these terms. At first, we compute the normal form of (12) modulo the ideal J , and then compare the coefficients of $x^i y^j$ of both sides to obtain the monomial vectors $m_{d_1}(\bar{X})$ and $m_{d_2}(\bar{X})$ which exploit the sparsity structure.

- The normal form of two sides of (12) modulo the ideal J

$$\begin{aligned} xy + 1 + y^2 + \epsilon = & w_{1,1} \quad v_{1,1} + v_{1,1}M \\ & + (w_{2,1} + w_{1,2} \quad v_{2,1} + v_{2,1}M \quad v_{1,2} + v_{1,2}M)x \\ & + (w_{3,5} + w_{5,3} \quad v_{3,4} \quad v_{2,1} + v_{2,6}M + w_{6,2} \\ & \quad v_{1,2} + v_{3,5}M + w_{2,6} \quad v_{2,5} + v_{1,3}M \quad v_{4,3} \\ & \quad v_{5,2} + w_{3,1} + w_{1,3} + v_{3,1}M + v_{5,3}M + v_{6,2}M)y \\ & + (w_{1,4} + w_{4,1} \quad v_{4,1} + v_{4,1}M \quad v_{2,2} + v_{2,2}M \quad v_{1,4} \\ & \quad + w_{2,2} + v_{1,4}M)x^2 + (v_{3,2}M + v_{6,4}M + w_{5,5} + w_{4,6} \\ & \quad + v_{2,3}M + v_{5,5}M + w_{2,3} \quad v_{2,2} + w_{1,5} + w_{3,2} \\ & \quad + v_{4,6}M \quad v_{1,4} + w_{6,4} + v_{5,1}M \quad v_{5,4} + v_{1,1} \\ & \quad + w_{5,1} + v_{1,5}M \quad v_{4,5} \quad v_{4,1})xy + (v_{3,3}M + v_{6,1}M \\ & \quad v_{2,3} + w_{6,5} \quad v_{5,5} \quad v_{4,6} + w_{6,1} + w_{1,6} \quad v_{1,1} \\ & \quad + w_{3,3} \quad v_{3,2} \quad v_{1,5} + v_{1,6}M \quad v_{5,1} \quad v_{6,4} + v_{6,5}M \\ & \quad + v_{5,6}M + w_{5,6})y^2 + (w_{4,2} \quad v_{2,4} + v_{4,2}M \quad v_{4,2} \\ & \quad + v_{2,4}M + w_{2,4})x^3 + (v_{2,4} + w_{4,3} + v_{5,2}M + v_{1,2} \\ & \quad + v_{3,4}M + w_{3,4} + v_{2,1} + v_{4,3}M + w_{2,5} \quad v_{4,2} + v_{2,5}M \\ & \quad + w_{5,2})x^2y + (w_{3,6} + w_{6,3} \quad v_{3,5} \quad v_{2,6} \quad v_{3,1} \\ & \quad + v_{6,3}M \quad v_{6,2} \quad v_{5,3} \quad v_{1,3} + v_{3,6}M)y^3 \\ & + (v_{4,4} + v_{4,4}M + w_{4,4})x^4 \\ & + (w_{5,4} + v_{2,2} + v_{1,4} + v_{4,1} + v_{4,5}M + v_{5,4}M \\ & \quad + w_{4,5} \quad v_{4,4})x^3y + (v_{6,5} \quad v_{6,1} \quad v_{5,6} \quad v_{3,3} \\ & \quad + v_{6,6}M \quad v_{1,6} + w_{6,6})y^4 + (v_{4,2} + v_{2,4})x^4y \\ & + (v_{6,3} \quad v_{3,6})y^5 \quad v_{6,6}y^6 + v_{4,4}x^5y. \end{aligned}$$

- The coefficients of y^6 and x^5y are $-v_{6,6}$ and $v_{4,4}$ respectively. Therefore $v_{4,4} = v_{6,6} = 0$. The matrix V is positive semidefinite, we have $v_{4,i} = v_{i,4} = v_{6,i} = v_{i,6} = 0$ for $1 \leq i \leq 6$.

- The coefficient of x^4 is $-v_{4,4} + v_{4,4}M + w_{4,4}$, we have $w_{4,4} = 0$. Since W is also positive semidefinite, we have $w_{i,4} = w_{4,i} = 0$ for $1 \leq i \leq 6$. From the coefficients of x^3y and x^2 , we can obtain that $v_{2,2} = w_{2,2} = 0$ and $v_{2,i} = v_{i,2} = w_{2,i} = w_{i,2} = 0$ for $1 \leq i \leq 6$.

- After eliminating all zero terms obtained above, we have

$$\begin{aligned} xy + 1 + y^2 + \epsilon = & w_{1,1} \quad v_{1,1} + v_{1,1}M \\ & + (w_{3,5} + w_{5,3} + v_{3,5}M + v_{1,3}M + w_{3,1} \\ & + w_{1,3} + v_{3,1}M + v_{5,3}M)y + (w_{5,5} + v_{5,5}M)xy \\ & + (v_{3,3}M + w_{6,5} \quad v_{5,5} + w_{6,1} + w_{1,6} \quad v_{1,1} \\ & + w_{3,3} \quad v_{1,5} \quad v_{5,1} + w_{5,6})y^2 \\ & + (w_{3,6} + w_{6,3} \quad v_{3,5} \quad v_{3,1} \quad v_{5,3} \\ & \quad v_{1,3})y^3 + (v_{3,3} + w_{6,6})y^4. \end{aligned}$$

- Deleting all zero rows and columns, one gets the simplified Gram matrices

$$W = \begin{bmatrix} w_{1,1} & w_{1,3} & w_{1,5} & w_{1,6} \\ w_{3,1} & w_{3,3} & w_{3,5} & w_{3,6} \\ w_{5,1} & w_{5,3} & w_{5,5} & w_{5,6} \\ w_{6,1} & w_{6,3} & w_{6,5} & w_{6,6} \end{bmatrix},$$

$$V = \begin{bmatrix} v_{1,1} & v_{1,3} & v_{1,5} \\ v_{3,1} & v_{3,3} & v_{3,5} \\ v_{5,1} & v_{5,3} & v_{5,5} \end{bmatrix}$$

corresponding to $m_{d_1}(\bar{X}) = [1, y, xy, y^2]$ and $m_{d_2}(\bar{X}) = [1, y, xy]$ respectively.

- For $M = 5$, the moment matrices corresponding to $m_{d_1}(\bar{X})$ and $m_{d_2}(\bar{X})$ are

$$\begin{bmatrix} y_{0,0} & y_{0,1} & y_{1,1} & y_{0,2} \\ y_{0,1} & y_{0,2} & y_{1,2} & y_{0,3} \\ y_{1,1} & y_{1,2} & y_{2,2} & y_{1,3} \\ y_{0,2} & y_{0,3} & y_{1,3} & y_{0,4} \end{bmatrix},$$

$$\begin{bmatrix} 4y_{0,0} + y_{1,1} - y_{0,2} & 5y_{0,1} - y_{0,3} & 5y_{1,1} - y_{0,2} \\ 5y_{0,1} - y_{0,3} & 5y_{0,2} - y_{0,4} & 5y_{0,1} - y_{0,3} \\ 5y_{1,1} - y_{0,2} & 5y_{0,1} - y_{0,3} & 5y_{1,1} - y_{0,2} \end{bmatrix}.$$

We can see that these moment matrices only consist of terms $y_{i,j}$ for $i \leq j$ which will go to 1 ($i = j$) or 0 ($i < j$) when xy goes to 1 and y goes to 0. Therefore the elements of the moment matrices which may cause the ill-conditionedness are removed.

For $k = 2, M = 5, A = I_{2 \times 2}, M_1 = 1000, M_2 = 1000$, the matrices W and V computed by our SDP solver in Maple for Digits = 60 are

$$W = \begin{bmatrix} 0.50804 & 0.0 & 0.0 & -0.50804 \\ 0.0 & 0.33126 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.13374 & 0.0 \\ -0.50804 & 0 & 0.0 & 0.50804 \end{bmatrix},$$

$$V = \begin{bmatrix} 0.12298 & 0.0 & -0.12298 \\ 0.0 & 0.13374 & 0.0 \\ -0.12298 & 0.0 & 0.12298 \end{bmatrix}.$$

The associated moment matrices are

$$\begin{bmatrix} 1 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}, \quad \begin{bmatrix} 5.0 & 0.0 & 5.0 \\ 0.0 & 0.0 & 0.0 \\ 5.0 & 0.0 & 5.0 \end{bmatrix}.$$

The lower bound we get is $f_2^* \approx 4.029500408 \times 10^{-24}$. Moreover, by SDPTools in Maple [2], we can obtain the certified lower bound

$$f_2^{**} = -4.029341206383157355520229568612510632 \times 10^{-24}$$

by writing $f - f_2^{**}$ as an exact rational SOS over W_M^A [7, 8].

EXAMPLE 5.3. Consider the following polynomial

$$f(x, y) = 2y^4(x + y)^4 + y^2(x + y)^2 + 2y(x + y) + y^2.$$

As mentioned in [3], we have $f^* = -\frac{5}{8}$ and f does not attain its infimum. It is also observed in [3] that there are obviously numerical problems since the output of their algorithm are $f_0^* = -0.614, f_1^* = -0.57314, f_2^* = -0.57259$, and $f_3^* = -0.54373$.

In fact, we have $f^* = f^{\text{SOS}} = -\frac{5}{8}$ since

$$\begin{aligned} f + \frac{5}{8} &= \frac{(2y^2 + 2xy + 1)^2(2y^2 + 2xy - 1)^2}{8} \\ &\quad + \frac{(2y^2 + 2xy + 1)^2}{2} + y^2. \end{aligned}$$

If we take $x_n = -(\frac{1}{n} + \frac{n}{2})$, $y_n = \frac{1}{n} - \frac{1}{n^3}$, it can be verified that $-\frac{5}{8}$ is a generalized critical value of f . For $k = 4$, if we do not exploit the sparsity structure, and choose

$$m_{d_1}(\bar{X}) = m_{d_2}(\bar{X}) := [1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3, x^4, x^3y, x^2y^2, xy^3, x^4]^T,$$

then numerical problems will appear.

By exploiting the sparsity structure of the SOS problem, we get

$$m_{d_1}(\bar{X}) = m_{d_2}(\bar{X}) := [1, y, y^2, xy, y^3, xy^2, y^4, xy^3, x^2y^2]^T,$$

the terms which cause ill-conditionedness of the moment matrix are removed. The lower bound computed by our SDP solver in Maple is $f_4^* = -0.625000000000073993$. It is very close to the true infimum -0.625 .

6. CONCLUSIONS

We use important properties in the computation of generalized critical values of a polynomial mapping [18, 19] and Theorem 1.3 to give a method to solve optimization (1). We do not require that f attains the infimum in $\mathbb{R}^n$ and use a much simpler variety in the SOS representation. We try to investigate and fix the numerical problems involved in computing the infimum of polynomials in Example 5.1 and 5.3. The strategies we propose here are just a first try. We hope to present a more general method to overcome these numerical problems in future.

Acknowledgments

We thank Markus Schweighofer for showing us [3], and the reviewers for their helpful comments.

7. REFERENCES

- [1] G. Blekherman. There are significantly more nonnegative polynomials than sums of squares. *Israel Journal of Mathematics*, 153(1):355–380, December 2006.
- [2] F. Guo. SDPTools: A high precision SDP solver in Maple. *MM-Preprints*, 28:66–73, 2009. Available at <http://www.mmrc.iss.ac.cn/mmreprints>.
- [3] H. V. Hà and T. S. Pham. Global optimization of polynomials using the truncated tangency variety and sums of squares. *SIAM J. on Optimization*, 19(2):941–951, 2008.
- [4] D. Henrion and J. B. Lasserre. GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi. *ACM Trans. Math. Softw.*, 29(2):165–194, 2003.
- [5] D. Jeffrey, editor. *ISSAC 2008*, New York, N. Y., 2008. ACM Press.
- [6] D. Jibeteau and M. Laurent. Semidefinite approximations for global unconstrained polynomial optimization. *SIAM J. on Optimization*, 16(2):490–514, 2005.
- [7] E. Kaltofen, B. Li, Z. Yang, and L. Zhi. Exact certification of global optimality of approximate factorizations via rationalizing sums-of-squares with floating point scalars. In Jeffrey [5], pages 155–163.
- [8] E. Kaltofen, B. Li, Z. Yang, and L. Zhi. Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients, 2009. Accepted for publication in *J. Symbolic Comput.*
- [9] K. Kurdyka, P. Orro, and S. Simon. Semialgebraic sard theorem for generalized critical values. *J. Differential Geom.*, 56(1):67–92, 2000.
- [10] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. on Optimization*, 11(3):796–817, 2001.
- [11] A. Lax and P. D. Lax. On sums of squares. *Linear Algebra and its Applications*, 20:71–75, 1978.
- [12] J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proc. IEEE CCA/ISIC/CACSD Conf.*, Taipei, Taiwan, 2004. Available at <http://users.isy.liu.se/johanl/yalmip/>.
- [13] Y. Nesterov. Squared functional systems and optimization problems. In H. Frenk, K. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, pages 405–440. Kluwer Academic Publishers, 2000.
- [14] J. Nie, J. Demmel, and B. Sturmfels. Minimizing polynomials via sum of squares over the gradient ideal. *Mathematical Programming*, 106(3):587–606, May 2006.
- [15] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. SOSTOOLS: Sum of squares optimization toolbox for MATLAB. 2004. Available at <http://www.cds.caltech.edu/sostools>.
- [16] B. Reznick. Extremal PSD forms with few terms. *Duke Mathematical Journal*, 45(2):363–374, 1978.
- [17] B. Reznick. Some concrete aspects of Hilbert’s 17th problem. In *In Contemporary Mathematics*, pages 251–272. American Mathematical Society, 1996.
- [18] M. Safey El Din. Testing sign conditions on a multivariate polynomial and applications. *Mathematics in Computer Science*, 1(1):177–207, 2007.
- [19] M. Safey El Din. Computing the global optimum of a multivariate polynomial over the reals. In Jeffrey [5], pages 71–78.
- [20] M. Safey El Din and É. Schost. Properness defects of projections and computation of at least one point in each connected component of a real algebraic set. *Discrete and Computational Geometry*, 32(3):417–430, September 2004.
- [21] M. Schweighofer. Global optimization of polynomials using gradient tentacles and sums of squares. *SIAM J. on Optimization*, 17(3):920–942, 2006.
- [22] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11/12:625–653, 1999.
- [23] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.
- [24] H. Waki, S. Kim, M. Kojima, M. Muramatsu, and H. Sugimoto. Algorithm 883: SparsePOP—a sparse semidefinite programming relaxation of polynomial optimization problems. *ACM Trans. Math. Softw.*, 35(2):1–13, 2008.

A Slice Algorithm for Corners and Hilbert-Poincaré Series of Monomial Ideals

Bjarke Hammersholt Røne
Department of Computer Science, Aarhus University
IT-Parken, Aabogade 34
8200 Aarhus N, Denmark
<http://www.broune.com/>

ABSTRACT

We present an algorithm for computing the corners of a monomial ideal. The corners are a set of multidegrees that support the numerical information of a monomial ideal such as Betti numbers and Hilbert-Poincaré series. We show an experiment using corners to compute Hilbert-Poincaré series of monomial ideals with favorable results.

Categories and Subject Descriptors

G.4 [Mathematics of Computing]: Mathematical Software; G.2.1 [Mathematics of Computing]: Discrete Mathematics—Combinatorial algorithms

General Terms

Algorithms, Performance

Keywords

Corners, Euler characteristic, Hilbert-Poincaré series, Koszul simplicial complex, monomial ideals

1. INTRODUCTION

We present an algorithm that computes the corners of a monomial ideal along with their Koszul simplicial complexes. This allows to compute Hilbert-Poincaré series, irreducible decomposition [5] and Koszul homology (as described e.g. in [7]).

In a sense the corners are those places on a monomial ideal where something “interesting” happens, and the Koszul simplicial complex for a corner encodes the local information about precisely what is happening there. In asking a computational (or otherwise) question about monomial ideals it is then a reasonable instinct to think about whether knowing the corners and their Koszul simplicial complexes would aid in answering that question. In this way corners could be a valuable tool in constructing algorithms, and the theoretical and practical value of the tool depends on the theoretical and practical performance of algorithms for corners.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

Dave Bayer introduced the concept of corners in 1996 [1]. Since then the first advance in this direction is a theoretical reverse search algorithm [2] for corners. As a reverse search algorithm it computes the corners of a monomial ideal in no more space up to a constant factor than that required by the input and output and in polynomial time.

Our contribution in this paper is an algorithm for corners that shows good practical performance. We demonstrate this by comparing it to the best algorithm for computing Hilbert-Poincaré series. We have not yet determined the theoretical time complexity of our algorithm, though this is an issue that deserves attention.

We call our algorithm a slice algorithm because it is inspired by and similar to the Slice Algorithm for maximal standard monomials of a monomial ideal [11]. Parts of the algorithm require modification to allow computation of corners, especially the proofs, though in particular the proof of termination is unchanged because it concerns the properties of monomial ideals and slices and not what is actually being computed. The main new idea that allows the Slice Algorithm to be applied to corners is that corners of full support have special properties that allow them to satisfy the equations that the Slice Algorithm is based on while corners in general do not. Due to a preprocessing step, the algorithm still manages to compute all corners, including those that do not have full support.

We wish to thank Eduardo Saenz-de-Cabezón and Anna Maria Bigatti for helpful discussions on these topics.

2. BACKGROUND AND NOTATION

Let I be a monomial ideal in some polynomial ring with indeterminates $x_1, \dots, x_n$. Let $\mathbb{x} \stackrel{\text{def}}{=} x_1 \cdots x_n$. We write a monomial $x_1^{v_1} \cdots x_n^{v_n}$ as x^v where v is the *exponent vector*. The *colon* of two monomials is $x^u : x^v \stackrel{\text{def}}{=} x^{\max(u-v, 0)}$ and we will have frequent use of the function $\pi(m) \stackrel{\text{def}}{=} m : \mathbb{x}$.

We can only very briefly cover the needed concepts. We recommend [8] for a more in-depth introduction.

2.1 Monomial ideals

A monomial ideal is an ideal generated by monomials. Then a monomial ideal I has a unique minimal set of monic monomial generators $\min(I)$. The *least common multiple* of two monomials is $\text{lcm}(x^u, x^v) \stackrel{\text{def}}{=} x^{\max(u, v)}$ and the *greatest common denominator* is $\text{gcd}(x^u, x^v) = x^{\min(u, v)}$. The colon of a monomial ideal by a monomial is $I : m \stackrel{\text{def}}{=} \langle a \mid am \in I \rangle = \langle a : m \mid a \in \min(I) \rangle$.

We plot a monomial ideal in a diagram by the exponent

vectors of the monomials in the ideal, such as seen in Figure 1. The surface displayed in such a diagram is known as the *staircase surface*, and the monomials on it are those $m \in I$ such that $\pi(m) \notin I$.

Define the *lcm lattice* of a monomial ideal I by $\text{lat}(I) \stackrel{\text{def}}{=} \{\text{lcm}(M) \mid M \subseteq \min(I)\}$ with lcm as the join and gcd as the meet of the lattice. So e.g. $\text{lat}(\langle x^2, xy \rangle) = \{1, x^2, xy, x^2y\}$.

The $\mathbb{N}^n$ -graded Hilbert-Poincaré series of I is the possibly infinite sum of all monomials that are not in I . This sum can be written as a fraction with $(1 - x_1) \cdots (1 - x_n)$ in the denominator and a polynomial $H(I)$ in the numerator. When we talk of computing the Hilbert-Poincaré series of I in this paper we are talking about computing $H(I)$. There is also the more conventional total degree-graded Hilbert-Poincaré series, which is obtained by substituting $x_i \mapsto t$ for each variable in the $\mathbb{N}^n$ -graded Hilbert-Poincaré series.

A monomial ideal I is (weakly) *generic* [9] if whenever $x^u, x^v \in \min(I)$ and $u_i = v_i > 0$ for some i , then either $u = v$ or there is some third generator in $\min(I)$ that strictly divides $\text{lcm}(x^u, x^v)$.

2.2 Simplicial complexes

An (abstract) *simplicial complex* Δ is a set of finite sets that is closed with respect to subset, i.e. if $v \in \Delta$ and $u \subseteq v$ then $u \in \Delta$. The elements of Δ are called *faces*, and the inclusion-maximal faces are called *facets*. We write $\text{fac}(\Delta)$ for the set of facets.

The faces in this paper are all subsets of $\{x_1, \dots, x_n\}$. The product of a face v is then $\Pi v \stackrel{\text{def}}{=} \prod_{x_i \in v} x_i$ and the intersection of a set of faces V is $\cap V \stackrel{\text{def}}{=} \cap_{v \in V} v$.

The (upper) *Koszul simplicial complex* of a monomial ideal I at a monomial m is defined by

$$\Delta_m^I \stackrel{\text{def}}{=} \left\{ v \subseteq \{x_1, \dots, x_n\} \mid \frac{m}{\Pi v} \in I \right\},$$

where $\frac{m}{\Pi v} \notin I$ when Πv does not divide m . So for $I \stackrel{\text{def}}{=} \langle x^2, xy \rangle$ we see that $\Delta_{x^2y}^I = \{\emptyset, \{x\}, \{y\}\}$ and $\Delta_{x^2}^I = \{\emptyset\}$.

We remark that Δ_m^I encodes the shape of the staircase surface of I around m . This yields interesting information about I at m , e.g. Δ_m^I determines the Betti numbers at m .

2.3 Corners

A monomial m is a *corner* of a monomial ideal I when no variable lies in every facet of Δ_m^I . The set of corners is then

$$\text{cor}(I) \stackrel{\text{def}}{=} \left\{ \text{monomials } m \mid \cap \text{fac}(\Delta_m^I) = \emptyset \right\}.$$

We do not consider m to be a corner if $\Delta_m^I = \emptyset$, while m is a corner if $\Delta_m^I = \{\emptyset\}$. So e.g. $\text{cor}(\langle x^2, xy \rangle) = \{x^2, xy, x^2y\}$.

The corners can be identified from a diagram of a monomial ideal as those points where the staircase surface is bent in every axis direction. The reader may verify that the corners lie on both the lcm lattice and the staircase surface.

As pointed out in [2], all multidegrees that have homology are corners. So knowing the set of corners and their Koszul simplicial complexes allows to determine interesting information such as the Betti numbers of I .

3. THE SLICE ALGORITHM

The Slice Algorithm we present here is a divide and conquer algorithm that computes the corners of a monomial ideal along with their Koszul simplicial complexes.

As a divide and conquer algorithm, the Slice Algorithm breaks the problem it is solving into two problems that are more easily solved. This process continues recursively until the problems are *base cases*, i.e. they are easy enough that they can be solved directly. The minimal ingredients of the Slice algorithm are then a recursive step, a base case, a proof of termination and a proof of correctness. In this section we present these along with pseudo code of the algorithm and an example of running the algorithm on a concrete ideal.

3.1 The recursive step

The Slice Algorithm operates on what we call *slices*. A slice A represents a subset of the corners of the input ideal, and we refer to this subset as the *content* $\text{con}(A)$ of the slice. The Slice Algorithm recursively splits a slice A into two less complicated slices B and C such that $\text{con}(A)$ is the disjoint union of $\text{con}(B)$ and $\text{con}(C)$.

We first present the formal definition of a slice and the equation we use to split slices. We follow that by an example that suggests a visual intuition of what the equation is stating. After that we prove that the equation is correct.

Definition 1. A *slice* is a 3-tuple (I, S, q) where I and S are monomial ideals and q is a monomial. The *content* of (I, S, q) is defined by

$$\text{con}(I, S, q) \stackrel{\text{def}}{=} \left\{ (mq, \Delta_{m\mathbb{x}}^I) \mid m\mathbb{x} \in \text{cor}(I) \text{ and } m \notin S \right\}.$$

The Slice Algorithm computes content, and this suffices to compute the set of corners, including those of non-full support, since $\text{cor}(I) = \text{con}(I\mathbb{x}, \langle 0 \rangle, 1)$. Note how the multiplication by $\mathbb{x}$ in $I\mathbb{x}$ and in the definition of content cancel each other out. This might seem to be a superfluous complication that we could resolve by simply removing $\mathbb{x}$ in both places. However, the significance of $\mathbb{x}$ in the definition of content is that we consider only corners of full support. Corners of full support have special properties that the Slice Algorithm depends on. This can be seen by the fact that many of our lemmas impose a condition of full support and that those lemmas cease to hold if the condition is lifted.

If C is a set of pairs (m, Δ) and S is a monomial ideal, then it will be of considerable convenience for us to perform set operations between C and S while not paying attention to the simplicial complexes of C . I.e.

$$\begin{aligned} C \cap S &\stackrel{\text{def}}{=} \{(m, \Delta) \in C \mid m \in S\}, \\ C \setminus S &\stackrel{\text{def}}{=} \{(m, \Delta) \in C \mid m \notin S\}. \end{aligned}$$

The Slice Algorithm uses the following equation to split a slice into two less complicated slices. We illustrate this in Example 1 and we discuss it further after the example.

$$\text{con}(I, S, q) = \text{con}(I : p, S : p, qp) \cup \text{con}(I, S + \langle p \rangle, q).$$

Example 1. Let $I := \langle x^6, x^5y^2, x^2y^4, y^6 \rangle$ and $p := xy^3$. Then I is the ideal depicted in Figure 1(a) where $\langle p \rangle$ is indicated by the dotted line. The corners are indicated by squares, and the squares for the corners of full support are filled. The full support corners are

$$\{x^2y^6, x^2y^4, x^5y^4, x^5y^2, x^6y^2\}.$$

We compute this set of full support corners by performing a step of the Slice Algorithm. We will not mention the Koszul

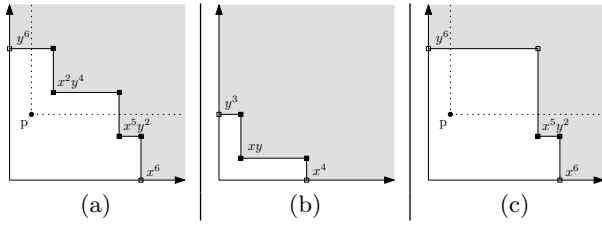


Figure 1: Illustrations for example 1.

simplicial complexes, but the reader may verify that these work out correctly as well.

Let I_1 be the ideal $I : p = \langle y^3, xy, x^4 \rangle$, as depicted in Figure 1(b). As can be seen by comparing figures 1(a) and 1(b), the ideal I_1 corresponds to the part of the ideal I that lies within $\langle p \rangle$. Thus it is reasonable to expect that the full support corners of I_1 correspond (after multiplication by p) to the full support corners of I that lie within $\langle p \rangle$. This turns out to be true, since

$$\{xy^3, xy, x^4y\} * p = \{x^2y^6, x^2y^4, x^5y^4\}.$$

It now only remains to find the full support corners of I that lie outside of $\langle p \rangle$. Let $I_2 := \langle x^6, x^5y^2, y^6 \rangle$ as depicted in Figure 1(c). The dotted line indicates that we are ignoring everything inside $\langle p \rangle$. It happens to be that one of the minimal generators of I , namely x^2y^4 , lies in the interior of $\langle p \rangle$, which allows us to ignore that minimal generator. We see that the corners of full support of I_2 that lie outside of $\langle p \rangle$ are $\{x^5y^2, x^6y^2\}$.

We have now found all the full support corners of I from the full support corners of I_1 and those full support corners of I_2 that lie outside of $\langle p \rangle$. Using the language of slices we have split the slice $A := (I, \langle 0 \rangle, 1)$ into the two slices $A_1 := (I_1, \langle 0 \rangle, p)$ and $A_2 := (I_2, \langle p \rangle, 1)$, and indeed

$$\begin{aligned} \text{con}(A) &= \{x^2y^6, x^2y^4, x^5y^4, x^5y^2, x^6y^2\} \\ &= (\{xy^3, xy, x^4y\} * xy^3) \cup \{x^5y^2, x^6y^2\} \\ &= \text{con}(A_1) \cup \text{con}(A_2), \end{aligned}$$

where the union is disjoint.

What we did in Example 1 was to rewrite $\text{con}(I, S, q)$ as $\text{con}(I, S, q) = (\text{con}(I, S, q) \cap \langle qp \rangle) \cup (\text{con}(I, S, q) \setminus \langle qp \rangle)$, and we wrote the two disjoint sets on the right hand side of this equation as the content of two slices. We now seek a way to do this given a general slice (I, S, q) and a monomial p . This is easy to do for the second set on the right hand side, since the definition of content implies that

$$\text{con}(I, S + \langle p \rangle, q) = \text{con}(I, S, q) \setminus \langle qp \rangle.$$

For the first set on the right hand side, we refer to Theorem 1, which states that

$$\text{con}(I : p, S : p, qp) = \text{con}(I, S, q) \cap \langle qp \rangle.$$

Example 1 gives an intuition of why this should be true.

Putting together the pieces, we get the *pivot split equation*

$$\text{con}(I, S, q) = \text{con}(I : p, S : p, qp) \cup \text{con}(I, S + \langle p \rangle, q). \quad (1)$$

This equation is the basic engine of the Slice Algorithm. We will discuss it and its parts at length, so we introduce names

to make such discussion convenient. The process of applying the pivot split equation is called a *pivot split* and p is the *pivot*. The left hand side slice (I, S, q) is the *current slice*, since it is the slice we are currently splitting. The first right hand slice $(I : p, S : p, qp)$ is the *inner slice*, since its content is inside $\langle qp \rangle$. The second right hand slice $(I, S + \langle p \rangle, q)$ is the *outer slice*, since its content is outside $\langle qp \rangle$.

We have stated that the Slice Algorithm splits a slice into two less complicated slices. So both the inner slice and the outer slice should be less complicated than the current slice. This is so for the inner slice because $I : p$ generally is a less complicated monomial ideal than I is. It is not immediately clear that the outer slice $(I, S + \langle p \rangle, q)$ is less complicated than the current slice. To see how it can be less complicated, consider Equation (2) which we prove in Theorem 1.

$$\text{cor}(I) \setminus S = \text{cor}(I') \setminus S, \quad I' \stackrel{\text{def}}{=} \langle m \in \min(I) \mid \pi(m) \notin S \rangle. \quad (2)$$

This equation states that we can remove from $\min(I)$ those elements that are strictly divisible by some element of S without changing the content of the slice. The outer slice has $S + \langle p \rangle$ where the current slice has S , so there is the potential to remove elements of $\min(I)$ due to Equation (2).

We apply Equation (2) whenever it is of benefit to do so, which it is when $\pi(\min(I)) \cap S \neq \emptyset$. Otherwise we say that the slice is *normal*, i.e. when $\pi(\min(I)) \cap S = \emptyset$.

THEOREM 1. *If p is a monomial, then*

$$i) \text{con}(I : p, S : p, qp) = \text{con}(I, S, q) \cap \langle qp \rangle,$$

$$ii) \text{con}(I, S, q) = \text{con}(I', S, q),$$

where $I' \stackrel{\text{def}}{=} \langle m \in \min(I) \mid \pi(m) \notin S \rangle$.

PROOF. *i)*: We get from the definition of content that

$$\begin{aligned} \text{con}(I : p, S : p, qp) &= \left\{ \left(mpq, \Delta_{m\mathbf{x}}^{I:p} \right) \mid m\mathbf{x} \in \text{cor}(I : p) \text{ and } m \notin S : p \right\} \\ &= \left\{ \left(m'q, \Delta_{m'\mathbf{x}:p}^{I:p} \right) \mid \begin{array}{l} p \text{ divides } m' \text{ and} \\ m' : p \notin S : p \text{ and} \\ m'\mathbf{x} : p \in \text{cor}(I : p) \end{array} \right\}, \\ \text{con}(I, S, q) \cap \langle qp \rangle &= \left\{ \left(mq, \Delta_{m\mathbf{x}}^{I:p} \right) \mid p \text{ divides } m \notin S \text{ and } m\mathbf{x} \in \text{cor}(I) \right\}. \end{aligned}$$

We prove that the two sets are equal by showing that each pair of similar conditions above are in fact equivalent. Even though m and m' are the same monomial, we retain the distinction to make it clear which set we are referring to. Going from left to right, we get by Lemma 1 that $\Delta_{m\mathbf{x}}^{I:p} = \Delta_{m'\mathbf{x}:p}^{I:p}$. This leaves the conditions to the right of the bar.

Whether $m\mathbf{x}$ is an element of $\text{cor}(I)$ depends only on $\Delta_{m\mathbf{x}}^{I:p}$. Likewise, whether $m'\mathbf{x} : p$ is an element of $\text{cor}(I : p)$ depends only on $\Delta_{m'\mathbf{x}:p}^{I:p}$. We have just seen that these two simplicial complexes are equal, so $m\mathbf{x}$ is an element of $\text{cor}(I)$ if and only if $m'\mathbf{x} : p$ is an element of $\text{cor}(I : p)$. This leaves only the matter of $m \notin S$ being equivalent to $m' : p \notin S : p$.

If t is a monomial such that $p \mid t$ then $t \in S \Leftrightarrow t : p \in S : p$, so $m \notin S$ if and only if $m' : p \notin S : p$ and we are done.

ii): Lemma 3 implies the more general statement that if $\pi(I) \setminus S = \pi(I') \setminus S$ then $\text{con}(I, S, q) = \text{con}(I', S, q)$. The former equation is satisfied by the particular I and I' in the

theorem since it holds for monomials a that

$$\begin{aligned} a \in \pi(I') \setminus S &\Leftrightarrow \exists m \in \min(I') : \pi(m) | a \text{ and } a \notin S \\ &\Leftrightarrow \exists m \in \min(I) : \pi(m) \notin S \text{ and } \pi(m) | a \text{ and } a \notin S \\ &\Leftrightarrow \exists m \in \min(I) : \pi(m) | a \text{ and } a \notin S \Leftrightarrow a \in \pi(I) \setminus S. \quad \square \end{aligned}$$

LEMMA 1. If $p|m$ then $\Delta_{m\mathbb{x}}^I = \Delta_{m\mathbb{x}:p}^{I:p}$.

PROOF. We use Lemma 2 with $A \stackrel{\text{def}}{=} I$, $B \stackrel{\text{def}}{=} (I:p)p$ and $c \stackrel{\text{def}}{=} m\mathbb{x}$. The preconditions of Lemma 2 are satisfied since $\langle \pi(m\mathbb{x}) \rangle = \langle m \rangle \subseteq \langle p \rangle$ so $A \cap \langle m \rangle = B \cap \langle m \rangle$. Then

$$\Delta_{m\mathbb{x}}^I = \Delta_{m\mathbb{x}}^{(I:p)p} = \Delta_{(m\mathbb{x}:p)p}^{(I:p)p} = \Delta_{m\mathbb{x}:p}^{I:p}. \quad \square$$

LEMMA 2. If A and B are monomial ideals and c is a monomial such that $A \cap \langle \pi(c) \rangle = B \cap \langle \pi(c) \rangle$, then $\Delta_c^A = \Delta_c^B$.

PROOF. Let $v \in \Delta_c^A$. Then $\pi(c) | \frac{c}{\Pi v} \in A$ so $\frac{c}{\Pi v} \in A \cap \langle \pi(c) \rangle = B \cap \langle \pi(c) \rangle$ so $\frac{c}{\Pi v} \in B$ so $v \in \Delta_c^B$. Swap A and B in this proof to get the other inclusion. $\square$

LEMMA 3. If A , B and C are monomial ideals such that $\pi(A) \setminus C = \pi(B) \setminus C$ and $m \notin C$ is a monomial, then $\Delta_{m\mathbb{x}}^A = \Delta_{m\mathbb{x}}^B$.

PROOF. Let $v \in \Delta_{m\mathbb{x}}^A$. Then $\frac{m\mathbb{x}}{\Pi v} \in A$ so $\frac{m}{\Pi v} \in \pi(A)$. As $\frac{m}{\Pi v} | m \notin C$ this implies that $\frac{m}{\Pi v} \in \pi(A) \setminus C = \pi(B) \setminus C$. Then $\frac{m}{\Pi v} \in \pi(B)$ so $\frac{m\mathbb{x}}{\Pi v} \in \pi(B)\mathbb{x} = B \cap \langle \mathbb{x} \rangle \subseteq B$ so $v \in \Delta_{m\mathbb{x}}^B$. Swap A and B in this proof to get the other inclusion. $\square$

3.2 The base case

In this section we present the base case of the Slice Algorithm. A slice (I, S, q) is a *base case slice* if I is square free or if I does not have full support (i.e. $\mathbb{x}$ does not divide $\text{lcm}(\min(I))$). Theorem 2 and Theorem 3 show how to obtain the content of a base case slice.

THEOREM 2. If I is a monomial ideal that does not have full support, then $\text{con}(I, S, q) = \emptyset$.

PROOF. No element of the lcm lattice of I has full support when I does not have full support. The corners of I lie on the lcm lattice of I , and the only corners of I we consider for the content are those of full support. $\square$

Recall that ϕ maps sets $v \subseteq \{x_1, \dots, x_n\}$ to the product of variables not in v , i.e. $\phi(v) = \Pi \bar{v} = \Pi_{x_i \notin v} x_i$. The main fact to keep in mind about ϕ is that it maps a subset relation into a domination relation, i.e. $v \supseteq u \Leftrightarrow \phi(v) | \phi(u)$.

THEOREM 3. If (I, S, q) is a slice such that I is square free and has full support, then $\text{con}(I, S, q) = \{(q, \Delta_{\mathbb{x}}^I)\}$ where $\text{fac}(\Delta_{\mathbb{x}}^I) = \phi^{-1}(\min(I))$.

PROOF. Lemma 4 implies that

$$\phi(\text{fac}(\Delta_{\mathbb{x}}^I)) = \min(I\mathbb{x} : \mathbb{x}) \setminus \langle x_1^2, \dots, x_n^2 \rangle = \min(I).$$

This implies that

$$\phi(\cap \text{fac}(\Delta_{\mathbb{x}}^I)) = \text{lcm}(\phi(\text{fac}(\Delta_{\mathbb{x}}^I))) = \text{lcm}(\min(I)) = \mathbb{x}.$$

Then $\cap \text{fac}(\Delta_{\mathbb{x}}^I) = \phi^{-1}(\mathbb{x}) = \emptyset$ so $\mathbb{x}$ is a corner of I . The corners of I lie on the lcm lattice, so they are all square free. We only consider corners of full support for the content, so $\mathbb{x}$ is the only corner that appears in the content. $\square$

LEMMA 4. If m is a monomial, then

$$\phi(\text{fac}(\Delta_m^I)) = \min(I\mathbb{x} : m) \setminus \langle x_1^2, \dots, x_n^2 \rangle.$$

PROOF. We see that $\phi(\text{fac}(\Delta_m^I)) = \min(\phi(\Delta_m^I))$. Then the result follows by applying $\min$ to both sides of

$$\phi(\Delta_m^I) = \{a \in I\mathbb{x} : m \mid a \text{ is a square free monomial}\}.$$

Every square free monomial can be written as $\phi(v)$ for some $v \subseteq \{x_1, \dots, x_n\}$, so this equation follows from

$$\begin{aligned} \phi(v) \in \phi(\Delta_m^I) &\Leftrightarrow v \in \Delta_m^I \Leftrightarrow \frac{m}{\Pi v} \in I \\ &\Leftrightarrow \frac{m\mathbb{x}}{\Pi v} \in I\mathbb{x} \Leftrightarrow m\phi(v) \in I\mathbb{x} \\ &\Leftrightarrow \phi(v) \in I\mathbb{x} : m. \quad \square \end{aligned}$$

3.3 Termination

We present four conditions on the choice of the pivot in pivot splits that are necessary and jointly sufficient to ensure termination. Each condition is independent of the others.

The conditions are listed below, along with an explanation of why violating any one of the conditions results in an inner or outer slice that is equal to the current slice. Once that happens the split can be repeated forever so that the Slice Algorithm would not terminate, so this shows that each condition is necessary. Note that just the first two conditions are sufficient to ensure termination at this point, but the last two conditions will become necessary after some of the improvements in Section 4 are applied.

Condition 1: $p \notin S$

Otherwise $p \in S$ and then the outer slice will be equal to the current slice.

Condition 2: $p \neq 1$

Otherwise $p = 1$ and then the inner slice will be equal to the current slice.

Condition 3: $p \notin I$

Otherwise the outer slice will be equal to the current slice after ‘‘Pruning of S ’’ from Section 4.

Condition 4: $p | \pi(\text{lcm}(\min(I)))$

Otherwise the outer slice will be equal to the current slice after ‘‘More pruning of S ’’ from Section 4.

We say that a pivot is *valid* when it satisfies these four conditions. Having imposed these conditions, we need to show that every slice that is not a base case admits a valid pivot (Theorem 4), and that it is not possible to keep splitting on valid pivots forever (Theorem 5).

THEOREM 4. If (I, S, q) is normal and admits no valid pivot, then I is square free and so (I, S, q) is a base case.

PROOF. Suppose I is not square free. Then there exists an x_i such that $x_i^2 | m$ for some $m \in \min(I)$, which implies that $x_i \notin I$. Also, $x_i \notin S$ since $x_i | \pi(m)$ and (I, S, q) is normal. We conclude that x_i is a valid pivot. $\square$

THEOREM 5. Selecting valid pivots ensures termination.

PROOF. The polynomial ring we are working within is noetherian, i.e. it does not contain an infinite sequence of ideals that is strictly increasing. We show that if the Slice Algorithm does not terminate, then such a sequence exists.

Let f and g be functions mapping slices to ideals, and define them by $f(I, S, q) \stackrel{\text{def}}{=} S$ and $g(I, S, q) \stackrel{\text{def}}{=} \langle \text{lcm}(\min(I)) \rangle$.

Suppose we split a non-base case slice A where A_1 is the inner slice and A_2 is the outer slice. Then Condition 1, Condition 2 and the fact that I has full support imply that

$$\begin{aligned} f(A) &\subseteq f(A_1), & g(A) &\subsetneq g(A_1), \\ f(A) &\subsetneq f(A_2), & g(A) &\subseteq g(A_2). \end{aligned}$$

Also, if we let A be an arbitrary slice and we let A' be the corresponding normal slice, then

$$f(A) \subseteq f(A'), \quad g(A) \subseteq g(A').$$

So we see that f and g never decrease, and one of them strictly increases on the outer slice while the other strictly increases on the inner slice. Thus there does not exist an infinite sequence of splits on valid pivots. $\square$

3.4 Pivot selection

In Section 3.3 we describe criteria on the selection of pivots that ensure that the algorithm completes its computation in some finite number of steps. It is good for the number of steps to be small rather than just finite, and for that the strategy used to select pivots plays an important role.

Our paper on the original Slice Algorithm for maximal standard monomials includes a section that proposes a number of different pivot selection strategies. We then compared all of these strategies to determine which one was the best over all. Due to space constraints we cannot present such an analysis here, though surely we will do so in a future journal version of this article.

We have still investigated the issue, and we can report that the pivot selection strategy that worked best for the previous Slice Algorithm remains competitive when computing corners. That strategy selects a pivot of the form x_i^e where x_i is a variable that maximizes $|\min(I) \cap \langle x_i \rangle|$ and e is the median exponent of x_i among the elements of $\min(I) \cap \langle x_i \rangle$. This kind of pivot selection strategy was first suggested by Anna Bigatti [3] in the context of the Bigatti et.al. algorithm for Hilbert-Poincaré series [4].

3.5 Pseudo code

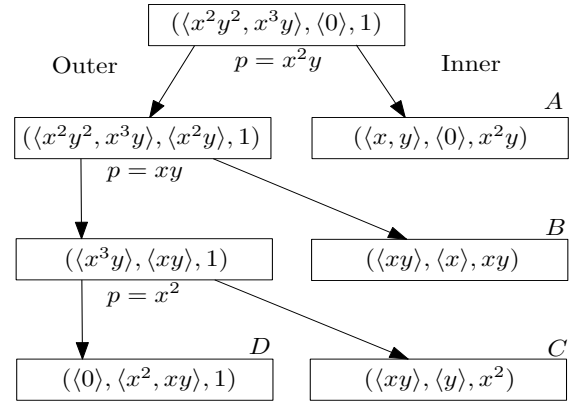
We show the Slice Algorithm in pseudo code.

```
function con(I, S, q)
  let I'  $\stackrel{\text{def}}{=} \langle m \in \min(I) \mid \pi(m) \notin S \rangle$ 
  if  $\mathbb{x}$  does not divide  $\text{lcm}(\min(I'))$  then return  $\emptyset$ 
  if I' is square free then return  $\{(q, \phi^{-1}(\min(I')))\}$ 
  let p be some monomial such that  $1 \neq p \notin S$ 
  return  $\text{con}(I' : p, S : p, qp) \cup \text{con}(I', S + \langle p \rangle, q)$ 
```

We have represented the simplicial complexes by their facets, so $\text{con}(I\mathbb{x}, \langle 0 \rangle, 1)$ returns $\{(m, \text{fac}(\Delta_m^I)) \mid m \in \text{cor}(I)\}$.

3.6 Example

The tree shows the steps of the algorithm on $\langle xy, x^2 \rangle$.



The contents of the leaves are (we specify facets only)

$$\begin{aligned} \text{con}(A) &= \{(x^2y, \{\{y\}, \{x\}\})\}, & \text{con}(B) &= \{(xy, \{\emptyset\})\}, \\ \text{con}(C) &= \{(x^2, \{\emptyset\})\}, & \text{con}(D) &= \emptyset. \end{aligned}$$

4. IMPROVEMENTS

In this section we show a number of improvements to the basic version of the Slice Algorithm presented so far.

It is natural that more specific versions of the improvements presented here also apply to the Slice Algorithm for maximal standard monomials and irreducible decomposition [11]. We use this fact in reverse by transferring the improvements to that algorithm to our current setting of corners and Koszul simplicial complexes. The improvements that rely only on the properties of monomial ideals and slices apply without change, while those that rely in their essence on the particular definition of content have to be adapted.

We summarize and classify each improvement according to whether it needs to be adapted. We refer to [11] for more detail on those improvements that apply without change.

Monomial lower bounds on slice contents: It is possible to replace a slice by a simpler slice with the same content using a monomial lower bound on the content. This improvement relies on the definition of content and so has to be adapted to apply to our setting.

Independence splits: This improvement applies to monomial ideals that have independent sets of variables. This needs some adaptation, but space does not permit to include it here.

A base case for two variables: There is a base case for ideals in two variables. This improvement has to be adapted to our setting.

Pruning of S: If (I, S, q) is a slice, this improvement is to remove elements of $\min(S)$ that lie in I . This can speed things up in case $|\min(S)|$ becomes large. The improvement and its proof apply without change.

More pruning of S: If (I, S, q) is a slice, this improvement is to remove elements of $\min(S)$ that do not strictly divide $\text{lcm}(\min(I))$. A significant implication of this is that pivots that are pure powers can always be removed from S after normalization. The improvement and its proof apply without change.

Minimizing the inner slice: This is a general monomial ideal technique for fast calculation of colons and intersections of a general monomial ideal by a principal

monomial ideal. This applies to computing inner slices. The technique applies without change.

Reduce the size of exponents: This is a general monomial ideal technique for supporting arbitrary precision exponents in a way that is as fast as using native machine integers. The technique applies without change.

4.1 Monomial lower bounds on slice contents

Let l be a *monomial lower bound* on the slice (I, S, q) in the sense that $ql|c$ for all $c \in \text{con}(I, S, q)$. In a pivot split on l , we can then predict that the outer slice will be empty. So the Pivot Split Equation (1) specializes to

$$\text{con}(I, S, q) = \text{con}(I : l, S : l, ql), \quad (3)$$

which shows that we can get the effect of performing a split while only having to compute a single slice. This is only interesting if we can determine a lower bound of a slice without already knowing its content, which is what Theorem 6 does.

THEOREM 6. *If (I, S, q) is a slice, then*

$$l_{x_i} \stackrel{\text{def}}{=} \pi(\gcd(\min(I) \cap \langle x_i \rangle))$$

is a monomial lower bound on (I, S, q) for each variable x_i .

PROOF. Suppose $c \in \text{cor}(I)$ such that $x_i|c$. As c lies on the lcm lattice there is then an $m \in \min(I)$ such that $x_i|m|c$ and then $\gcd(\min(I) \cap \langle x_i \rangle)|m|c$.

If $qc \in \text{con}(I, S, q)$ then $c\mathbf{x} \in \text{cor}(I)$, and we have just proven that this implies that $l_{x_i} = \pi(\cdots)|\pi(c\mathbf{x}) = c$. $\square$

Theorem 6 allows us to make a slice simpler with no change to the content, and this can be iterated until a fixed point is reached simultaneously for every variable.

4.2 A base case of two variables

If $n = 2$ then the corners and their simplicial Koszul complexes can be computed directly at only the cost of sorting the minimal generators. This can be relevant even if the input ideal is in more than two variables since independence splits generate slices in fewer variables than the input.

Let $\min(I) = \{m_1, \dots, m_k\}$ where $m_1, \dots, m_k$ are sorted in ascending lexicographic order with $x_1 > x_2$. There are only two kinds of corners for $n = 2$. The first are the generators $a_1, \dots, a_k$, and the Koszul simplicial complex for all of these is $\{\emptyset\}$.

The second kind of corner are the maximal staircase monomials. Let $\psi(x^u, x^v) \stackrel{\text{def}}{=} x_1^{v_1} x_2^{u_2}$. Then the maximal staircase monomials are $\psi(a_1, a_2), \dots, \psi(a_{k-1}, a_k)$. These all have complex $\{\emptyset, \{x_1\}, \{x_2\}\}$.

Example 2. For $I \stackrel{\text{def}}{=} \langle xy^5, x^2y, x^5 \rangle$ we have

$$\begin{aligned} \text{con}(I\mathbf{x}, \langle 0 \rangle, 1) = & \{ \langle xy^5, \{\emptyset\} \rangle, \langle x^2y, \{\emptyset\} \rangle, \langle x^5, \{\emptyset\} \rangle, \\ & \langle x^2y^5, \{\emptyset, \{x_1\}, \{x_2\} \rangle \rangle, \\ & \langle x^5y, \{\emptyset, \{x_1\}, \{x_2\} \rangle \rangle \}. \end{aligned}$$

5. HILBERT-POINCARÉ SERIES

In this section we describe how to use corners and Koszul simplicial complexes to compute Hilbert-Poincaré series. We do so based on a formula for the Hilbert-Poincaré series that is due to Dave Bayer [1, Proposition 3.2]. This formula uses the *Euler characteristic* of a simplicial complex.

Definition 2. The *Euler characteristic* of Δ is defined by

$$\chi(\Delta) \stackrel{\text{def}}{=} \sum_{v \in \Delta} (-1)^{|v|-1}.$$

The formula that we use is then

$$H(I) - 1 = \sum_{v \in \mathbb{N}^n} \chi(\Delta_{x^v}^I) x^v = \sum_{m \in \text{cor}(I)} \chi(\Delta_m^I) m. \quad (4)$$

Recall that $H(I)$ is the numerator of the Hilbert-Poincaré series of I . So from this formula we see that we can determine the Hilbert-Poincaré series of I from the corners of I and their Koszul simplicial complexes, and that the way we do so is by computing the Euler characteristic of each complex. We call this the Corner-Euler Algorithm. The Slice Algorithm provides the corners and their complexes, so the only missing part is how to compute the Euler characteristic of a simplicial complex.

6. EULER CHARACTERISTIC

In this section we present a new algorithm for computing Hilbert-Poincaré series. We give the big picture of how the algorithm works as space permits. Since coming up with this algorithm we have found that very little work has been done on this topic, so we have since investigated the area in much more detail in upcoming joint work with Eduardo Saenz-de-Cabezon [12].

To compute the Euler characteristic, we are going to use a characterization in terms of the square free ideal $\langle \phi(\Delta) \rangle$. Since $\Delta_{\mathbf{x}}^{\langle \phi(\Delta) \rangle} = \Delta$, we get from Equation 4 that

$$\text{Coefficient of } \mathbf{x} \text{ in } H(\langle \phi(\Delta) \rangle) = \chi(\Delta_{\mathbf{x}}^{\langle \phi(\Delta) \rangle}) = \chi(\Delta).$$

Thus computing the Euler characteristic of a simplicial complex amounts to computing the coefficient of $\mathbf{x}$ in $H(I)$ for I a square free monomial ideal. In this way it makes sense to define $\chi(I)$ as the coefficient of $\mathbf{x}$ in $H(I)$.

We could compute all of $H(I)$ to get $\chi(I)$, but we don't have to. The divide and conquer algorithm by Bigatti et.al. [4, 3] is the best known way to compute Hilbert-Poincaré series. It is based on repeated application of the equation

$$H(I) = H(I : p)p + H(I + \langle p \rangle), \quad (5)$$

where p is a monomial. For square free p this implies that

$$\chi(I) = \chi(I : p) + \chi(I + \langle p \rangle),$$

where we embed $I : p$ in the subring of the ambient polynomial ring that excludes those variables that divide p .

This equation suggests a divide and conquer algorithm for computing the Euler characteristic of simplicial complexes. One base case occurs when I does not have full support since then $\chi(I) = 0$. The other base case occurs when I has full support and the elements of $\min(I)$ are relatively prime, since then $\chi(I) = (-1)^{|\min(I)|}$. A good choice of p is $p = x_i$ where x_i maximizes $|\min(I) \cap \langle x_i \rangle|$.

We use this algorithm to implement the Euler characteristic computation step of the Corner-Euler Algorithm. Running this algorithm for the Koszul simplicial complex of every corner might seem like it would take a lot of time, but in fact in our implementation it generally takes longer to compute the corners and Koszul simplicial complexes in the first place.

We should point out that the Corner-Euler Algorithm for computing Hilbert-Poincaré series is not equivalent to the Bigatti et.al. Algorithm, even though the Euler characteristic computation is also based on Equation 5. One way to see this is to consider that there can be corners of $I + \langle p \rangle$ and corners of $I : p$ that are not corners of I . The Bigatti et.al. Algorithm can tolerate this because any terms of $H(I : p)p$ and $H(I + \langle p \rangle)$ that correspond to these additional corners will cancel out such that they do not appear in the final output. In contrast the Slice Algorithm looks only for the actual corners.

Since no terms cancel in the output of the Corner-Euler Algorithm, it is possible to output a term and vacate it from memory as soon as it is computed. In contrast the Bigatti et.al. Algorithm has to wait for the extra terms to cancel, and so the terms that occur in the Hilbert-Poincaré series numerator are not identifiable until the end of the computation.

7. EXPERIMENTS

In this section we gauge the practical performance of the algorithms in this paper. Unfortunately, we know of no serious implementations of algorithms for corners that we might compare ours against. The computation of Hilbert-Poincaré series has, however, received a lot of attention both in the literature and in terms of being implemented, and so we look at the Corner-Euler Algorithm for Hilbert-Poincaré series for this experiment, as an indirect way of examining the performance of the Slice Algorithm.

These experiments show that the Corner-Euler Algorithm as presented here is a reasonable algorithm for computing Hilbert-Poincaré series, and that in some cases it is even faster than the Bigatti et.al. Algorithm. It reflects well on the Slice Algorithm for corners and Koszul simplicial complexes that it can compute what it does fast enough that it can be any sort of a competitor to the Bigatti et.al. Algorithm, given that that algorithm is the best result from the effort that has gone into research on Hilbert-Poincaré series computation from a number of prominent authors. We conclude from that that the Slice Algorithm is practical and thus that corners can be used as a practical tool in monomial ideal computations.

We have implemented both the Corner-Euler Algorithm and the Bigatti et.al. Algorithm in the software system Frobby [10], which is an open source and freely available system for monomial ideal computations. These implementations are of comparable quality and written by the same person to make the comparison as fair as possible. The implementation of the Bigatti et.al. Algorithm in CoCoA [6] is the leading implementation, so we include that in the comparison as well.

We employ a suite of ten ideals for the experiment, and we name them respectively generic, nongeneric, squarefree and toric. These ideals have been selected from a long list of possible ideals that we could have used. The ideals are among those attached to the web version of [11]. They have been selected on the basis of providing interesting information and for being neither trivial nor so demanding that the experiment will run for too a long time. Table 1 has further information. It would be wonderful to use a more extensive suite of examples, as we will surely do in a future journal version of this article, but space does not permit it here.

generic: These ideals have been randomly generated with exponents in the range $[0, 30000]$. The ideals are thus very close to generic.

nongeneric: These ideals have been randomly generated with exponents in the range $[0, 10]$. The ideals are thus far from generic and also far from being square free.

squarefree: These ideals have been randomly generated with exponents in the range $[0, 1]$. They are thus square free and farthest from generic.

toric: This ideal is the initial ideal of a toric ideal defined by a primitive vector with eight entries that are random numbers of 30 decimal digits each. The ideal is generic and has exponents in the range $[0, 95998]$. Computing the Hilbert-Poincaré series of this ideal is a subalgorithm in computing the genus of the numerical semigroup generated by the primitive vector.

We run two experiments, one for computing the $\mathbb{N}^n$ -graded Hilbert-Poincaré series, and the other for the conventional total degree-graded Hilbert-Poincaré series. These are shown in Table 2 and 3 respectively. We have been in contact with the authors of CoCoA, but have so far been unable to make computing multigraded Hilbert-Poincaré series work in CoCoA. We are still investigating how to make this work.

We take away from Table 1 that in general most corners do show up in the multigraded Hilbert-Poincaré series. This is good news for the Corner-Euler algorithm since it is based on computing all the corners.

One conclusion we can draw is that the Corner-Euler Algorithm is faster for the multigraded computation than for the univariate one. This is because the Corner-Euler Algorithm can output terms as soon as they are computed in the former case, but in the latter case it is necessary to collect like terms before output and this takes extra time.

The Bigatti et.al. Algorithm has the opposite behavior, being faster for the univariate computation than the multivariate one except for the inputs with high exponents. This is because univariate computations allow a base case that is very fast when the degrees of the generators are not too high. Otherwise the base case is exponential in the number of variables, and avoiding this is part of the benefit that the Bigatti et.al. Algorithm derives from the univariate computation in low degrees.

We conjecture that the reason that the Corner-Euler Algorithm is faster than the Bigatti et.al. Algorithm for generic ideals is that in those cases the number of terms that the Bigatti et.al. Algorithm generates that are not actually part of the output is much higher than for other ideals. It would be interesting to count the number of superfluous intermediate terms to verify or reject this hypothesis.

8. REFERENCES

- [1] D. Bayer. Monomial ideals and duality. Never finished draft. See <http://www.math.columbia.edu/~bayer/vita.html>, 1996.
- [2] D. Bayer and A. Taylor. Reverse search for monomial ideals. *Journal of Symbolic Computation*, 44:1477–1486, 2009.
- [3] A. M. Bigatti. Computation of Hilbert-Poincaré series. *Journal of Pure and Applied Algebra*, 119(3):237–253, 1997.

name	n	$\min(I)$	terms of $H(I)$	corners
generic1	10	80	455,076	455,076
generic2	10	120	1,364,358	1,364,358
generic3	10	160	2,940,226	2,940,226
nongeneric1	10	100	83,867	117,635
nongeneric2	10	150	506,001	778,324
nongeneric3	10	200	796,931	1,256,896
squarefree1	20	1,000	81,704	105,037
squarefree2	20	2,000	142,384	173,075
squarefree3	20	4,000	251,650	299,788
toric	8	2,099	2,948,154	2,948,154

Table 1: Further information about the ideals.

software algorithm	Frobby Corner-E.	Frobby Bigatti ea.
generic1	2s	67s
generic2	6s	323s
generic3	13s	754s
nongeneric1	<1s	4s
nongeneric2	7s	25s
nongeneric3	11s	38s
squarefree1	12s	6s
squarefree2	28s	12s
squarefree3	75s	30s
toric	18s	133s

Table 2: Multigraded Hilbert-Poincaré series.

software algorithm	Frobby Corner-E.	Frobby Bigatti ea.	CoCoA4 Bigatti ea.
generic1	3s	79s	394s
generic2	8s	352s	1,078s
generic3	18s	793s	1,783s
nongeneric1	1s	3s	<1s
nongeneric2	8s	13s	<1s
nongeneric3	13s	21s	<1s
squarefree1	12s	<1s	<1s
squarefree2	28s	<1s	2s
squarefree3	74s	1s	4s
toric	23s	133s	514s

Table 3: Univariate Hilbert-Poincaré series.

- [4] A. M. Bigatti, P. Conti, L. Robbiano, and C. Traverso. A “divide and conquer” algorithm for Hilbert-Poincaré series, multiplicity and dimension of monomial ideals. In *Applied algebra, algebraic algorithms and error-correcting codes (San Juan, PR, 1993)*, volume 673 of *Lecture Notes in Comput. Sci.*, pages 76–88. Springer, Berlin, 1993.
- [5] A. M. Bigatti and E. S. de Cabezón. (n-1)-st Koszul homology and the structure of monomial ideals. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 31–38, New York, NY, USA, 2009. ACM.
- [6] CoCoATeam. CoCoA: a system for doing Computations in Commutative Algebra. Available at <http://cocoa.dima.unige.it>.
- [7] E. S. de Cabezón. Combinatorial Koszul homology: Computations and applications, 2008. <http://arxiv.org/abs/0803.0421>.
- [8] E. Miller and B. Sturmfels. *Combinatorial Commutative Algebra*, volume 227 of *Graduate Texts in Mathematics*. Springer, 2005.
- [9] E. Miller, B. Sturmfels, and K. Yanagawa. Generic and cogenerated monomial ideals. *Journal of Symbolic Computation*, 29(4-5):691–708, 2000. Available at <http://www.math.umn.edu/~ezra/papers.html>.
- [10] B. H. Roune. Frobby version 0.9 – a software system for computations with monomial ideals. Available at <http://www.broune.com/frobby/>.
- [11] B. H. Roune. The Slice Algorithm for irreducible decomposition of monomial ideals. *Journal of Symbolic Computation*, 44(4):358–381, April 2009.
- [12] B. H. Roune and E. S. de Cabezón. Combinatorial commutative algebra algorithms for the euler characteristic of abstract simplicial complexes. XII ENCuentro de Álgebra Computacional Y APLICACIONES (Spanish meeting on Computer Algebra and Applications), 2010.

Composition Collisions and Projective Polynomials

Statement of Results*

Joachim von zur Gathen
B-IT, Universität Bonn
D-53113 Bonn, Germany

gathen@bit.uni-bonn.de
<http://cosec.bit.uni-bonn.de/>

Mark Giesbrecht
Cheriton School of Computer Science
University of Waterloo, Waterloo, ON, N2L 3G1 Canada

mwg@cs.uwaterloo.ca
<http://www.cs.uwaterloo.ca/~mwg>

Konstantin Ziegler
B-IT, Universität Bonn
D-53113 Bonn, Germany

zieglerk@bit.uni-bonn.de
<http://cosec.bit.uni-bonn.de/>

ABSTRACT

The functional decomposition of polynomials has been a topic of great interest and importance in pure and computer algebra and their applications. The structure of compositions of (suitably normalized) polynomials $f = g \circ h$ in $\mathbb{F}_q[x]$ is well understood in many cases, but quite poorly when the degrees of both components are divisible by the characteristic p . This work investigates the decomposition of polynomials whose degree is a power of p . An (equal-degree) i -collision is a set of i distinct pairs (g, h) of polynomials, all with the same composition and $\deg g$ the same for all (g, h) . Abhyankar (1997) introduced the *projective polynomials* $x^n + ax + b$, where n is of the form $(r^m - 1)/(r - 1)$ and r is a power of p . Our first tool is a bijective correspondence between i -collisions of certain additive trinomials, projective polynomials with i roots, and linear spaces with i Frobenius-invariant lines.

Blüher (2004b) has determined the possible number of roots of projective polynomials for $m = 2$, and how many polynomials there are with a prescribed number of roots. We generalize her first result to arbitrary m , and provide an alternative proof of her second result via elementary linear algebra.

If one of our additive trinomials is given, we can efficiently compute the number of its decompositions, and similarly the number of roots of a projective polynomial. The runtime of these algorithms depends polynomially on the sparse input size, and thus on the input degree only logarithmically.

For non-additive polynomials, we present certain decompositions and conjecture that these comprise all of the prescribed shape.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

Categories and Subject Descriptors

F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—*Computations on polynomials*

General Terms

Theory

Keywords

Univariate polynomial decomposition, additive polynomials, linearized polynomials, projective polynomials

1. INTRODUCTION

The *composition* of two polynomials $g, h \in F[x]$ over a field F is denoted by $f = g \circ h = g(h)$, and then (g, h) is a *decomposition* of f . In the 1920s, Ritt, Fatou, and Julia studied structural properties of these decompositions over $\mathbb{C}$, using analytic methods. Particularly important are two theorems by Ritt on uniqueness, in a suitable sense, of decompositions, the first one for (many) indecomposable components and the second one for two components, as above.

The theory was algebraicized by Dorey & Whaples (1974), Schinzel (1982, 2000), and others. Its use in a cryptographic context was suggested by Cade (1985). In computer algebra, the method of Barton & Zippel (1985) requires exponential time but works in all situations. A breakthrough result of Kozen & Landau (1989) was their polynomial-time algorithm to compute decompositions. One has to distinguish between the *tame case*, where the characteristic p does not divide $\deg g$ and this algorithm works (see von zur Gathen (1990a)), and the *wild case*, where p divides $\deg g$ (see von zur Gathen (1990b)). In the wild case, considerably less is known, mathematically and computationally. The algorithm of Zippel (1991) for decomposing rational functions suggests that the block decompositions of Landau & Miller (1985) (for determining subfields of algebraic number fields) can be applied to the wild case. Giesbrecht (1998) provides fast algorithms for the decomposition of additive (or linearized) polynomials,

*Subtitle suggested by the Programme Committee Chair

in some sense an “extremely wild” case. We exploit their elegant structure here. An enumeration of number or structure of solutions in the wild case has defied both algebraic and computational analysis, and we attempt to address this here. Moreover, many of the algorithms we present here are sensitive to the sparse size of the input, as opposed to the degree, a property not exploited in the above-mentioned papers.

The task of counting compositions over a finite field of characteristic p was first considered in Giesbrecht (1988). Von zur Gathen (2009) presents general approximations to the number of decomposable polynomials. These come with satisfactory (rapidly decreasing) relative error bounds except when p divides $n = \deg f$ exactly twice. The goal of the present work is to study the easiest of these difficult cases, namely when $n = p^2$ and hence $\deg g = \deg h = p$. However, many of our results are valid for more general powers of p and stated accordingly.

We introduce the notion of an equal-degree i -collision of decompositions, which is a set of i pairs (g, h) , all with the same composition and $\deg g$ the same for all (g, h) . These are the only collisions we consider in this paper, and we omit the adjective “equal-degree” in the text. An i -collision is *maximal* if it is not contained in an $(i + 1)$ -collision. After some preliminaries in Section 2, we start in Section 3 with the particular case of additive polynomials. We relate the decomposition question to one about eigenspaces of the linear function given by the Frobenius map on the roots of f . This yields a complete description of all decompositions of certain additive trinomials in terms of the roots of the *projective polynomials* $x^n + ax + b$, introduced by Abhyankar (1997), where n is of the form $(r^m - 1)/(r - 1)$, for a power r of p . We prove that maximal i -collisions of additive polynomials of degree r^2 exist only when i is 0, 1, 2 or $r + 1$, count their numbers exactly, and show their relation to the roots of projective polynomials for $m = 2$. In this case Bluher (2004b) has determined, the number of roots that can occur, namely 0, 1, 2, or $r + 1$, and also for how many coefficients (a, b) each case happens. We obtain elementary proofs of a generalization of her first result to arbitrary m and of her counts for $m = 2$. From the proof we obtain a fast algorithm (polynomial in r and $\log q$) to count the number of roots over $\mathbb{F}_q$, called *rational* roots. More generally, in Section 4 an algorithm is provided to enumerate the possible number of right components of an additive polynomial of any degree. A fast algorithm is then presented to count the number of right components of an additive polynomial of any degree, which is shown to be equivalent to counting rational roots of projective polynomials of arbitrary degree. We also demonstrate theorems and fast algorithms to count and construct indecomposable additive polynomials of prescribed degree. In Section 5 we actually construct and enumerate all additive polynomials of degree r^2 with 0, 1, 2, or $r + 1$ collisions and establish connections to the counts of Bluher (2004b) and von zur Gathen (2009).

In Section 6 we move from additive to general polynomials. Certain $(r + 1)$ -collisions are derived from appropriate roots of projective polynomials. We conjecture that these are all possibilities and present results on general i -collisions with $i \geq 2$ for $r = p$ that support our conjecture.

Due to the page restriction, no proofs appear here. They can be found in the full version (von zur Gathen, Giesbrecht & Ziegler, 2010).

2. THE BASIC SETUP

We consider polynomials $f, g, h \in \mathbb{F}_q[x]$ over a finite field $\mathbb{F}_q$ of characteristic p . Then $f = g \circ h = g(h)$ is the *composition* of g and h , (g, h) is a *decomposition* of f , and g and h are a *left* and *right component*, respectively, of f . Furthermore, f is *decomposable* if such (g, h) exist with $\deg g, \deg h \geq 2$, and *indecomposable* otherwise.

We call f *original* if its graph passes through the origin, that is, if $f(0) = 0$. Composition with linear polynomials introduces inessential ambiguities in decompositions. If $f = g \circ h$, $a \in \mathbb{F}_q^\times$, and $b \in \mathbb{F}_q$, then $af + b = (ag + b) \circ h$. Thus we may assume f to be monic original. Furthermore, if $a = \text{lc}(h)^{-1}$ and $b = -ah(0)$, then $f = g \circ h = g((x - b)a^{-1}) \circ (ah + b)$ and the right component is monic original. Therefore we may also assume h to be monic original, and then g is so automatically. We thus consider the following two sets:

$$P_n(\mathbb{F}_q) = \{f \in \mathbb{F}_q[x] : f \text{ is monic and original of degree } n\}, \\ D_n(\mathbb{F}_q) = \{f \in P_n(\mathbb{F}_q) : f \text{ is decomposable}\}.$$

We usually leave out the argument $\mathbb{F}_q$. The size of the first set is $\#P_n = q^{n-1}$, and determining (exactly or approximately) $\#D_n$ is one of the goals in this business. The number of all or all decomposable polynomials of degree n , not restricted to P_n , is $\#P_n$ or $\#D_n$, respectively, multiplied by $q(q - 1)$.

First, we consider the additive or linearized polynomials, which have a mathematically rich and highly useful structure in finite fields. First introduced in Ore (1933), they play an important role in the theory of finite and function fields, and they have found many applications in codes and cryptography. See Lidl & Niederreiter (1983), Chapter 3, for an introduction and survey over finite fields.

We focus on additive polynomials over finite fields, though some of these results will hold more generally in characteristic p . We take a power r of p and a power q of r . Let

$$\mathbb{F}_q[x; r] = \left\{ \sum_{0 \leq i \leq m} a_i x^{r^i} : m \in \mathbb{Z}_{\geq 0}, a_0, \dots, a_m \in \mathbb{F}_q \right\}$$

be the ring of r -additive (or *linearized*, or simply *additive*) polynomials over $\mathbb{F}_q$. These are the polynomials such that $f(\alpha a + \beta b) = \alpha f(a) + \beta f(b)$ for any $\alpha, \beta \in \mathbb{F}_r$, and for any $a, b \in \mathbb{F}_q$, where $\mathbb{F}_r$ is an algebraic closure of $\mathbb{F}_q$. The additive polynomials form a (non-commutative) ring under the usual addition and composition. It is a principal left (and right) ideal ring with a left (and right) Euclidean algorithm.

An additive polynomial is squarefree if f' (the derivative of f) is nonzero, meaning that the linear coefficient of f is nonzero. If $f \in \mathbb{F}_q[x; r]$ is squarefree of degree r^m , then the set of all roots of f form an $\mathbb{F}_r$ -vector space in $\overline{\mathbb{F}_r}$ of dimension m . Conversely, for any finite dimensional $\mathbb{F}_r$ -vector space $W \subseteq \overline{\mathbb{F}_r}$, the lowest degree polynomial $f = \prod_{a \in W} (x - a) \in \mathbb{F}_r[x]$ with W as its roots is a squarefree r -additive polynomial. Let σ_q denote the q th power Frobenius automorphism on $\overline{\mathbb{F}_q}$ over $\mathbb{F}_q$. If W is invariant under σ_q , then $f \in \mathbb{F}_q[x; r]$.

We have

$$x^p \circ h = \sigma_p(h) \circ x^p$$

for $h \in \mathbb{F}_q[x]$, where σ_p is the Frobenius automorphism on $\mathbb{F}_q$ over $\mathbb{F}_p$, which extends to polynomials coefficientwise. If $\deg h = p$ and $h \neq x^p$, this is a 2-collision and called a *Frobenius collision*. It is never part of i -collisions with $i \geq 3$.

LEMMA 2.1. Let $S \in \mathbb{F}_r^{m \times m}$ be the matrix representing the Frobenius σ_q . There is a bijection between S -invariant subspaces of $\mathbb{F}_r^{m \times 1}$ and right components $h \in \mathbb{F}_q[x; r]$ of f .

We present two related approaches to investigate $f \in \mathbb{F}_q[x; r]$ of degree r^2 . The first, working with normal forms of the Frobenius operator on the space of roots of f , gives a straightforward classification of the number of possible decompositions, though provides less insight into how many polynomials fall into each class. The second uses more structural information about the ring of additive polynomials and provides complete information on both the number of decompositions and the number of polynomials with each type of decomposition.

A non-squarefree $f = x^{r^2} + ax^r \in \mathbb{F}_q[x; r]$ is a 2-collision if $a \neq 0$ and has a unique decomposition if $a = 0$.

Closely related to decompositions are the following objects. Let r be a power of p , $m \geq 1$, and $\varphi_{r,m} = (r^m - 1)/(r - 1)$. Abhyankar (1997) introduced the *projective polynomials*

$$\Psi_m^{(a,b)} = x^{\varphi_{r,m}} + ax + b$$

which have, over appropriate fields, nice Galois groups such as general linear or projective general linear groups. We assume q to be a power of r , and have for $m = 2$

$$\Psi_2^{(a,b)} = x^{r+1} + ax + b \quad (2.1)$$

with $a, b \in \mathbb{F}_q$. In the case $ab \neq 0$, Bluher (2004b) has proven an amazingly precise result about the number of nonzero roots of (2.1). Namely, this number is 0, 1, 2, or $r + 1$, and she has exactly determined the number of parameters (a, b) for which each of the four possibilities occurs. In the case $a = 0$, the corresponding number is given in von zur Gathen (2008), Lemma 5.9.

Projective polynomials appear naturally in many situations. Bluher (2004a) used them to construct strong Davenport pairs explicitly and Dillon (2002) to build families of difference sets with certain Singer parameters. Bluher (2003) proved the equivalence of two such difference sets, using again projective polynomials and they played a central role in tackling the question of when a quartic power series over $\mathbb{F}_q$ is actually hyperquadratic (Bluher & Lasjaunias, 2006).

Helleseth, Kholosha & Johanssen (2008) used projective polynomials to find m -sequences of length $2^{2k} - 1$ and $2^k - 1$. Helleseth & Kholosha (2010) studied projective polynomials further, providing criteria for the number of zeros in a field of characteristic 2, not assuming q to be a power of r . Zeng, Li & Hu (2008) applied the techniques of Bluher (2004b) to study certain p -ary codes.

3. ADDITIVE AND PROJECTIVE POLYNOMIALS

We assume that q is a power of r and r is a power of the characteristic p of $\mathbb{F}_q$. In this section we establish a general connection between decompositions of certain additive polynomials and roots of projective polynomials, and characterize the possible numbers of rational roots of the latter.

LEMMA 3.1. Let $m \geq 1$, $f = x^{r^m} + ax^r + bx$ and $h = x^r - h_0x$ be in $\mathbb{F}_q[x; r]$ with $a, b, h_0 \in \mathbb{F}_q$. Then $f = g \circ h$ for some $g \in \mathbb{F}_q[x; r]$ if and only if $\Psi_m^{(a,b)}(h_0) = 0$.

This lemma and Lemma 2.1 are the building blocks for the powerful equivalences summarized as follows.

PROPOSITION 3.2. Let r be a power of p , $m \geq 1$, $a, b \in \mathbb{F}_q$ and $f = x^{r^m} + ax^r + b$. There is a one-to-one correspondence between any two of the following sets.

- right components of f with degree r ,
- roots of $\Psi_m^{(a,b)}$,
- σ_q -invariant linear subspaces of V_f with dimension 1.

More generally, assume that $f \in \mathbb{F}_q[x; r]$ is any additive polynomial of degree r^m . We now list the possible numbers of right components in $\mathbb{F}_q[x; r]$. A *rational Jordan form* has the shape

$$S = \text{diag}(J_{\alpha_1}^{e_{11}}, \dots, J_{\alpha_1}^{e_{1k_1}}, \dots, J_{\alpha_\ell}^{e_{\ell 1}}, \dots, J_{\alpha_\ell}^{e_{\ell k_\ell}}) \in \mathbb{F}_r^{m \times m},$$

$$\text{where } J_{\alpha_i}^{e_{ij}} = \begin{pmatrix} C_{\alpha_i} I_{s_i} & 0 \\ & \ddots & \ddots \\ & & I_{s_i} \\ & & & C_{\alpha_i} \end{pmatrix} \in \mathbb{F}_r^{e_{ij}s_i \times e_{ij}s_i}, \quad (3.1)$$

and $\alpha_1, \dots, \alpha_\ell \in \overline{\mathbb{F}_r}$ are the distinct non-conjugate roots of the characteristic polynomial of S (i.e., eigenvalues), $C_{\alpha_i} \in \mathbb{F}_r^{s_i \times s_i}$ is the companion matrix of α_i (assuming $[\mathbb{F}_r[\alpha_i] : \mathbb{F}_r] = s_i$) and I_{s_i} is the $s_i \times s_i$ identity matrix.

Let V_f be the $\mathbb{F}_r$ -vector space of roots, and $S \in \mathbb{F}_r^{m \times m}$ the matrix representation of the Frobenius operation σ_q on $\overline{\mathbb{F}_r}$. It is well-known (see, e.g. Giesbrecht (1995)) that every matrix in $\mathbb{F}_r^{m \times m}$ is similar to one in rational Jordan form, and the number and multiplicity of eigenvectors is preserved by this transformation. Thus, we may assume S to be of the form described in (3.1). Since we are only interested here in σ_q -invariant subspaces of dimension 1, we ignore for now all α_i which are not in $\mathbb{F}_r$. The number of A -invariant lines — one dimensional subspaces invariant under A — is described as follows.

THEOREM 3.3. If $A \in \mathbb{F}_r^{m \times m}$ has rational Jordan normal form as in (3.1), then the number of A -invariant lines in $\mathbb{F}_r^{m \times 1}$ is

$$\sum_{\substack{1 \leq i \leq \ell \\ \alpha_i \in \mathbb{F}_r}} \varphi_{r, k_i}.$$

For example, in $\mathbb{F}_r^{3 \times 3}$ we can list all matrix classes and the number of 1-dimensional invariant subspaces as follows:

$$\begin{pmatrix} \alpha_1 & & \\ & \alpha_1 & \\ & & \alpha_1 \end{pmatrix}_{r^2+r+1}, \quad \begin{pmatrix} \alpha_1 & 1 & \\ & \alpha_1 & \\ & & \alpha_1 \end{pmatrix}_{r+1}, \quad \begin{pmatrix} \alpha_1 & 1 & \\ & \alpha_1 & 1 \\ & & \alpha_1 \end{pmatrix}_1, \quad \begin{pmatrix} \alpha_1 & 1 & \\ & \alpha_1 & \\ & & \alpha_2 \end{pmatrix}_2$$

$$\begin{pmatrix} \alpha_1 & & \\ & \alpha_1 & \\ & & \alpha_2 \end{pmatrix}_{r+2}, \quad \begin{pmatrix} \alpha_1 & & \\ & \alpha_2 & \\ & & \alpha_3 \end{pmatrix}_3, \quad \left(\begin{array}{c|c} \square & \\ \hline & \alpha_1 \end{array} \right)_1, \quad \left(\begin{array}{c|c} \square & \\ \hline & \end{array} \right)_0,$$

where the number of 1-dimensional invariant subspaces is listed beneath each matrix. Empty boxes indicate companion blocks associated with eigenvalues not in $\mathbb{F}_r$.

For a positive integer m , let Π_m be the set of partitions $\pi = (s_1, \dots, s_k)$ with positive integers s_i and $s_1 + \dots + s_k = m$, for any $\pi \in \Pi_m$, let $\varphi_r(\pi) = \varphi_{r,s_1} + \varphi_{r,s_2} + \dots + \varphi_{r,s_k}$ and $\varphi_r(\Pi_m) = \{\varphi_r(\pi) : \pi \in \Pi_m\}$.

THEOREM 3.4. We consider the set

$$S_{q,r,m} = \{i \in \mathbb{N} : \exists f \in \mathbb{F}_q[x; r], \deg f = r^m, \\ f \text{ has a maximal } i\text{-collision}\}$$

of maximal collision sizes for additive polynomials. Then

$$S_0 = \{0\}, \\ S_m = S_{m-1} \cup \varphi_r(\Pi_m).$$

As examples, we have

$$\begin{aligned} S_0 &= \{0\}, \\ S_1 &= S_0 \cup \{\varphi_r(1)\} = \{0, 1\}, \\ S_2 &= S_1 \cup \{\varphi_r(1, 1), \varphi_r(2)\} = \{0, 1, 2, r+1\}, \\ &\quad (\text{consistent with Bluher (2004b)}) \\ S_3 &= S_2 \cup \{\varphi_r(3), \varphi_r(2) + 1, 3\}, \\ S_4 &= S_3 \cup \{\varphi_r(4), \varphi_r(3) + 1, 2\varphi_r(2), \varphi_r(2) + 2, 4\}. \end{aligned}$$

The size of S_m equals $\sum_{0 \leq k \leq m} p(k)$, where $p(k)$ is the number of additive partitions of k . This grows exponentially in m (Hardy & Ramanujan, 1918) but is still surprisingly small considering the generality of the polynomials involved. By Proposition 3.2, S_m consists of the number of roots of any $\Psi_m^{(a,b)}$, and equivalently the number of σ_q -invariant linear subspaces of V_f of dimension 1 for any $f = x^{r^m} + ax^r + bx$.

We investigate the general result of Theorem 3.4 in the case $m = 2$ further. This leads, for each i , to an exact determination of how often i -collisions occur; consistent with Bluher (2004b). Assume that $f \in \mathbb{F}_q[x; r]$ is squarefree, with root space V_f . Again let σ_q be the Frobenius automorphism fixing $\mathbb{F}_q$, and $S \in \mathbb{F}_r^{2 \times 2}$ its representation with respect to some fixed basis. The number of one-dimensional subspaces of V_f invariant under σ_q is equal to the number of nonzero vectors $w \in \mathbb{F}_r^{2 \times 1}$ such that $Sw = \lambda w$ for some $\lambda \in \mathbb{F}_r$, that is, the number of eigenvalues of S . Each such w generates a one-dimensional σ_q -invariant subspace, and each such subspace is generated by $r - 1$ such w . Thus, the number of distinct σ_q -invariant subspaces of dimension one, and hence the number of right components in $\mathbb{F}_q[x; r]$ of degree r , is equal to the number of eigenvectors of S in $\mathbb{F}_r^2$, divided by $r - 1$.

We now classify σ_q according to the possible matrix similarity classes of S , as captured by its rational canonical form, and count the number of eigenvectors and components in each case. Note that the number of eigenvectors of S equals the number of eigenvectors of T when S is a similar matrix to T ($S \sim T$).

THEOREM 3.5. *Let $f \in \mathbb{F}_q[x; r]$ be squarefree of degree r^2 . Suppose the Frobenius automorphism σ_q is represented by $S \in \mathbb{F}_r^{2 \times 2}$, and $\Lambda \in \mathbb{F}_r[z]$ is the minimal polynomial of the matrix S . Then one of the following holds:*

Case 0: $S \sim \begin{pmatrix} 0 & \delta \\ 1 & \gamma \end{pmatrix}$, and $\Lambda = z^2 - \gamma z - \delta \in \mathbb{F}_r[z]$ is irreducible, and f is indecomposable.

Case 1: $S \sim \begin{pmatrix} \gamma & 1 \\ 0 & \gamma \end{pmatrix} \in \mathbb{F}_r^{2 \times 2}$ with $\gamma \neq 0$, and $\Lambda = (z - \gamma)^2$, and f has a unique right component of degree r .

Case 2: $S \sim \begin{pmatrix} \gamma & 0 \\ 0 & \delta \end{pmatrix} \in \mathbb{F}_r^{2 \times 2}$ for $\gamma \neq \delta$ with $\gamma\delta \neq 0$, when $\Lambda = (z - \gamma)(z - \delta)$, and f has a 2-collision.

Case $r + 1$: $S = \begin{pmatrix} \gamma & 0 \\ 0 & \gamma \end{pmatrix} \in \mathbb{F}_r^{2 \times 2}$, for $\gamma \neq 0$, and f has an $(r + 1)$ -collision.

4. ALGORITHMS FOR ADDITIVE POLYNOMIALS

Given $f \in \mathbb{F}_q[x; r]$ of degree r^2 , using the techniques of Section 3, combined with basic algorithms from Giesbrecht (1998), we can quickly determine the number of collisions for f . The centre of $\mathbb{F}_q[x; r]$ will be a useful tool in understanding

its structure, and is easily shown to be equal to

$$\mathbb{F}_r[x; q] = \left\{ \sum_{0 \leq i \leq \kappa} a_i x^{q^i} : \kappa \in \mathbb{Z}_{\geq 0}, a_0, \dots, a_\kappa \in \mathbb{F}_r \right\} \subseteq \mathbb{F}_q[x; r]$$

(see, e.g., Giesbrecht (1998)). This is isomorphic to the ring $\mathbb{F}_r[y]$ of polynomials under the usual addition and multiplication, via the isomorphism

$$f = \sum_{0 \leq i \leq \kappa} a_i x^{q^i} \mapsto \tau(f) = \sum_{0 \leq i \leq \kappa} a_i y^i$$

(see Lidl & Niederreiter (1983), Section 3.4). $\mathbb{F}_r[y]$ has the important property of being a commutative unique factorization domain. Every element $f \in \mathbb{F}_q[x; r]$ has a unique *minimal central left composition* (mclc) $f^* \in \mathbb{F}_r[x; q]$, the nonzero monic polynomial in $\mathbb{F}_r[x; q]$ of minimal degree such that $f^* = g \circ f$ for some $g \in \mathbb{F}_q[x; r]$. Given $\nu \in \mathbb{F}_r$, we say that ν belongs to $f \in \mathbb{F}_q[x; r]$ if f is the nonzero polynomial in $\mathbb{F}_q[x; r]$ of lowest degree of which ν is a root.

FACT 4.1 (GIESBRECHT, 1998). *Let p be a prime, r a power of p and $q = r^d$. For $f \in \mathbb{F}_q[x; r]$ of degree r^m , we can find the minimal central left composition $f^* \in \mathbb{F}_r[x; q]$ with $O(d^3 m^3)$ operations in $\mathbb{F}_r$.*

The following key theorem shows the close relationship between the minimal central left composition and the minimal polynomial of the Frobenius automorphism.

THEOREM 4.2. *Let $f \in \mathbb{F}_q[x; r]$ be squarefree of degree r^m with roots $V_f \subseteq \mathbb{F}_r$. Fix an $\mathbb{F}_r$ -basis $\mathcal{B} = \langle \nu_1, \dots, \nu_m \rangle \in \mathbb{F}_r^m$ for V_f , so that $V_f \cong \mathbb{F}_r^{m \times 1}$. Let $S \in \mathbb{F}_r^{m \times m}$ represent the action of the Frobenius automorphism σ_q on V_f with respect to $\mathcal{B}$. Then the image $\tau(f^*) \in \mathbb{F}_r[y]$ of the minimal central left composition $f^* \in \mathbb{F}_r[x; q]$ of f is equal to the minimal polynomial $\Lambda \in \mathbb{F}_r[x]$ of the matrix S .*

It is useful to recall a little more about the ring $\mathbb{F}_q[x; r]$. Ore (1933) shows that for any $f, g \in \mathbb{F}_q[x; r]$, there exists a unique monic $h \in \mathbb{F}_q[x; r]$ of maximal degree, and $u, v \in \mathbb{F}_q[x; r]$, such that $f = u \circ h$ and $g = v \circ h$, called the *greatest common right component* (gcr) of f and g . Also, $h = \text{gcr}(f, g) = \text{gcd}(f, g)$, and the roots of h are those in the intersection of the roots of g and h . Furthermore, there exists a unique monic and nonzero $h \in \mathbb{F}_q[x; r]$ of minimal degree, and $u, v \in \mathbb{F}_q[x; r]$, such that $h = u \circ f$ and $h = v \circ g$, called the *least common left composition* (lcl) of f, g . The roots of h are the $\mathbb{F}_r$ -vector space sum of the roots of f and g ; this sum is direct if $\text{gcr}(f, g) = 1$. In fact, there is an efficient Euclidean-like algorithm for computing the lcl and gcr; see, Ore (1933), and Giesbrecht (1998) for an analysis.

We now present our algorithm to count decompositions of polynomials in $\mathbb{F}_q[x; r]$ of degree r^2 .

ALGORITHM: DecompositionCounting

Input: $\triangleright f \in \mathbb{F}_q[x; r]$ of degree r^2 , where $q = r^d$

Output: $\triangleright$ The number of decompositions of f

- (1) If $f'(0) = 0$ Then
- (2) If $f = x^{r^2}$ Then Return 1
- (3) Else Return 2
- (4) Else $f^* \leftarrow \text{mclc}(f) \in \mathbb{F}_r[x; q]$
- (5) If $\deg f^* = r$ Then Return $r + 1$
- (6) Factor $\tau(f^*) \in \mathbb{F}_r[y]$ over $\mathbb{F}_r[y]$

- (7) If $\tau(f^*) \in \mathbb{F}_r[y]$ is irreducible Then Return 0
- (8) If $\tau(f^*) = (y - a)^2$ for some $a \in \mathbb{F}_r$ Then Return 1
- (9) Return 2

Well-known factorization methods yield the following.

THEOREM 4.3. *The algorithm **DecompositionCounting** works as specified and requires an expected number of $O(d^3) \log r$ operations in $\mathbb{F}_r$ using a randomized algorithm, or $d^{O(1)} \log r$ operations with a deterministic algorithm (assuming the ERH).*

The algorithm **DecompositionCounting** also yields the number of rational roots of the projective polynomial $x^{r+1} + ax + b$ (Proposition 3.2).

For the remainder of this section we look at the problem of counting the number of irreducible right components of degree r of any additive polynomial $f \in \mathbb{F}_q[x; r]$ of degree r^m . The algorithm will run in time polynomial in m and $\log q$. This will also yield a fast algorithm to compute the number of rational roots of a projective polynomial $\Psi_m^{(a,b)} \in \mathbb{F}_q[x]$.

The approach is to compute explicitly the Jordan form of the Frobenius operator σ_q acting on the roots of f , as in (3.1). We show how to do this quickly, despite the fact that the actual roots of f may lie in an extension of exponential degree over $\mathbb{F}_q$.

ALGORITHM: FindJordan

Input: $\blacktriangleright f \in \mathbb{F}_q[x; r]$ monic squarefree of degree r^m , where r is a prime power

Output: $\blacktriangleright$ Rational Jordan form $S \in \mathbb{F}_r^{m \times m}$ of the Frobenius automorphism $\sigma_q(a) = a^q$ (for $a \in \overline{\mathbb{F}_r}$) on V_f , as in (3.1)

- (1) Compute $f^* \leftarrow \text{mlc}(f) \in \mathbb{F}_r[x; q]$
- (2) Factor $\tau(f^*) \leftarrow u_1^{\omega_1} u_2^{\omega_2} \cdots u_\ell^{\omega_\ell} \in \mathbb{F}_r[y]$, where the $u_i \in \mathbb{F}_r[y]$ are monic irreducible and pairwise distinct, and $\deg u_i = s_i$ for $1 \leq i \leq \ell$
- (3) For i from 1 to ℓ do
- (4) For j from 1 to ω_i do
- (5) $h_{ij} \leftarrow \text{gcr}(\tau^{-1}(u_i^j), f)$
- (6) $\xi_{ij} \leftarrow (\log_r h_{ij}) / s_i$ (i.e., $\deg h_{ij} = r^{s_i \xi_{ij}}$)
- (7) For j from 1 to $\omega_i - 1$ do
- (8) $\delta_{ij} \leftarrow \xi_{ij} - \xi_{i,j+1}$
- (9) $\delta_{i\omega_i} \leftarrow \xi_{i\omega_i}$
- (10) $k_i \leftarrow \xi_{i1}$
- (11) $(e_{i1}, \dots, e_{ik_i}) \leftarrow (\underbrace{1, \dots, 1}_{\delta_{i1}}, \underbrace{2, \dots, 2}_{\delta_{i2}}, \dots, \underbrace{\omega_i, \dots, \omega_i}_{\delta_{i\omega_i}})$
- (12) Return $S = \text{diag} \left(J_{\alpha_1}^{e_{11}}, \dots, J_{\alpha_1}^{e_{1k_1}}, \dots, J_{\alpha_\ell}^{e_{\ell 1}}, \dots, J_{\alpha_\ell}^{e_{\ell k_\ell}} \right)$

THEOREM 4.4. *The algorithm **FindJordan** works as specified. It requires an expected number of operations in $\mathbb{F}_q$ which is polynomial in m and $\log r$ (Las Vegas).*

Now given an $f \in \mathbb{F}_q[x; r]$ we can quickly compute the rational Jordan form of the Frobenius automorphism on its root space. Computing the number of degree r factors (or indeed, the number of irreducible factors of any degree) is easy, following the same method as in Section 3.

THEOREM 4.5. *If the Frobenius automorphism of the root space of an $f \in \mathbb{F}_q[x; r]$ has rational Jordan form in the*

*notation of Algorithm **FindJordan** where*

$$S = \text{diag} \left(J_{\alpha_1}^{e_{11}}, \dots, J_{\alpha_1}^{e_{1k_1}}, \dots, J_{\alpha_\ell}^{e_{\ell 1}}, \dots, J_{\alpha_\ell}^{e_{\ell k_\ell}} \right),$$

$$(e_{i1}, \dots, e_{ik_i}) \leftarrow (\underbrace{1, \dots, 1}_{\delta_{i1}}, \underbrace{2, \dots, 2}_{\delta_{i2}}, \dots, \underbrace{\omega_i, \dots, \omega_i}_{\delta_{i\omega_i}})$$

for $1 \leq i \leq \ell$, then the number of indecomposable right components of degree r is

$$\sum_{s_i=1} \sum_{1 \leq j \leq \omega_i} \delta_{ij} \cdot \frac{r^j - 1}{r - 1}.$$

Thus, the number of right components of degree r of an additive polynomial of degree r^m can be computed in time polynomial in m and $\log q$, and also the number of roots in $\mathbb{F}_r$ of a projective polynomial (Lemma 3.1).

5. PROJECTIVE POLYNOMIALS AND ROOTS

We now actually construct and enumerate all the polynomials in each case 0, 1, 2, $r+1$ as in Theorem 3.5.

THEOREM 5.1. *Let r be a prime power and q a power of r . For $i \in \mathbb{N}$ let*

$$C_{q,r,m,i} = \{ (a, b) \in \mathbb{F}_q^2 : x^{r^m} + ax^r + bx \text{ has a maximal } i\text{-collision in } \mathbb{F}_q[x; r] \},$$

$$c_{q,r,m,i} = \#C_{q,r,m,i},$$

and drop q, r, m from the notation. For $m = 2$, the following holds:

Case 0: C_0 is the set of all $f \in \mathbb{F}_q[x; r]$ of degree r^2 whose minimal central left compositions $f^* \in \mathbb{F}_r[x; q]$ have degree q^2 and cannot be written as $f^* = g^* \circ h^*$ for $g^*, h^* \in \mathbb{F}_r[x; q]$ of degree q , or equivalently that the image $\tau(f^*) \in \mathbb{F}_r[y]$ of f^* is irreducible of degree 2. We have

$$c_0 = \frac{r(q^2 - 1)}{2(r + 1)}.$$

Case 1: C_1 is the set of all $f \in \mathbb{F}_q[x; r]$ of degree r^2 with minimal central left composition $f^* = g^* \circ g^*$ for $g^* = x^q - cx$ for $c \in \mathbb{F}_r^\times$, and

$$c_1 = \frac{q^2 - q}{r} + 1.$$

Case 2: C_2 is the set of all $f \in \mathbb{F}_q[x; r]$ with minimal central left composition $f^* = g^* \circ h^*$ for $g^*, h^* \in \mathbb{F}_r[x; q]$ of degree q with $\gcd(g^*, h^*) = 1$, and

$$c_2 = \frac{(q - 1)^2 \cdot (r - 2)}{2(r - 1)} + q - 1.$$

Case $r+1$: C_{r+1} is the set of all $f \in \mathbb{F}_q[x; r]$ of degree r^2 with minimal central left composition $f^* = x^q + cx$, for $c \in \mathbb{F}_r^\times$, and

$$c_{r+1} = \frac{(q - 1)(q - r)}{r(r^2 - 1)}.$$

Since $c_0 + c_1 + c_2 + c_{r+1} = q^2$, these are the only possible numbers of collisions of a degree r^2 polynomial in $\mathbb{F}_q[x; r]$.

In each case, the number of collisions of an $f \in \mathbb{F}_q[x; r]$ is determined by the factorization of its minimal central left composition $f^* \in \mathbb{F}_r[x; q]$. Here $\deg \tau(f^*) \in \{1, 2\}$, and we

can enumerate all such f^* in each class (irreducible linear, irreducible quadratic, perfect square, or product of distinct linear factors). We can decompose each such f^* using the algorithms of Giesbrecht (1998) to generate polynomials with a prescribed number of collisions.

We show now how to construct indecomposable additive polynomials of prescribed degree, and count their number. We also show how to construct additive polynomials with a single, unique complete decomposition and count the number of such polynomials.

The following theorem characterizes indecomposable polynomials of degree r^ℓ in terms of their minimal central left compositions. This theorem allows us to get hold of degree r right components from the roots of $\tau(f^*)$ in $\mathbb{F}_q$.

THEOREM 5.2 (GIESBRECHT, 1998, THEOREM 4.3). *Let $f^* \in \mathbb{F}_r[x; q]$ have degree q^ℓ , such that $\tau(f^*) \in \mathbb{F}_r[y]$ is irreducible (of degree ℓ). Then every indecomposable right component $f \in \mathbb{F}_q[x; r]$ of f^* has degree r^ℓ . Conversely, all $f \in \mathbb{F}_q[x; r]$ which are indecomposable of degree r^ℓ are such that $\tau(f^*) \in \mathbb{F}_r[y]$ is irreducible of degree ℓ , where $f^* \in \mathbb{F}_r[x; q]$ is the minimal central left composition of f .*

The following bound has been shown in Odoni (1999). Our methods here provide a simple proof. Let

$$I_r(n) = \sum_{d|n} \mu(n/d) r^d$$

be the number of monic irreducible polynomials in $\mathbb{F}_r[y]$ of degree n (see, e.g., Lidl & Niederreiter (1983), Theorem 3.25).

THEOREM 5.3. *Let q be a power of r . The number of monic indecomposable polynomials $f \in \mathbb{F}_q[x; r]$ of degree r^m is*

$$\frac{q^m - 1}{r^m - 1} I_r(m).$$

This implies there are (slightly) more indecomposable additive polynomials of degree r^m in $\mathbb{F}_q[x; r]$ than irreducible polynomials of degree m in $\mathbb{F}_q[y]$.

The above theorem also yields a reduction from the problem of finding indecomposable polynomials in $\mathbb{F}_q[x; r]$ of prescribed degree to that of decomposing polynomials in $\mathbb{F}_q[x; r]$. A fast randomized algorithm for decomposing additive polynomials is shown in Giesbrecht (1998), which requires a number of operations bounded by $(m + \log q)^{O(1)}$. Thus, we can just choose a random polynomial in $\mathbb{F}_q[x; r]$ of prescribed degree and check if it is irreducible, with a high expectation of success. A somewhat slower polynomial-time reduction from decomposing additive polynomials in $\mathbb{F}_q[x; r]$ to factoring in $\mathbb{F}_r[y]$ is also given in Giesbrecht (1998). This suggests the interesting question as to whether one can find indecomposable polynomials in $\mathbb{F}_q[x; r]$ of prescribed degree n in deterministic polynomial-time, assuming the ERH (à la Adleman & Lenstra (1986)).

We finish this section by establishing connections to the counts of Bluher (2004b) and von zur Gathen (2009). Proposition 3.2 yields an equivalent description of $C_{q,r,m,i}$ as

$$C_{q,r,m,i} = \{(a, b) \in \mathbb{F}_q^2 : \Psi_m^{(a,b)} \text{ has exactly } i \text{ roots in } \mathbb{F}_q\}.$$

Section 3 says that

$$C_{q,r,m,i} \neq \emptyset \implies i \in S_{q,r,m}$$

and $S_{q,r,m}$ is determined in Theorem 3.4. Furthermore, let

$$\begin{aligned} C_{q,r,m,i}^{(1)} &= \{(a, b) \in C_{q,r,m,i} : b \neq 0\}, \\ C_{q,r,m,i}^{(2)} &= \{(a, b) \in C_{q,r,m,i} : ab \neq 0\}, \end{aligned}$$

and $c_{q,r,m,i}^{(j)} = \#C_{q,r,m,i}^{(j)}$ for $j = 1, 2$. Leaving out the indices, we have $C^{(2)} \subseteq C^{(1)} \subseteq C$. The set $C^{(1)}$ occurs naturally in general decompositions (Proposition 6.5 (iii) for $r = p$), and $C^{(2)}$ is the subject of Bluher (2004b). For an integer $m \geq 1$, let

$$\gamma_{q,r,m} = \gcd(\varphi_{r,m}, q - 1).$$

PROPOSITION 5.4. *We fix q, r, m as above and drop them from the notation of $C_{q,r,m,i}^{(j)}$ and $c_{q,r,m,i}^{(j)}$.*

(i) *We have $C_i = C_i^{(1)}$ for all $i \notin \{1, \gamma_{m-1} + 1\}$, and*

$$\begin{aligned} C_1 \setminus C_1^{(1)} &= \{(a, 0) : (-a)^{(q-1)/\gamma_{q,r,m-1}} \neq 1\}, \\ C_{\gamma_{m-1}+1} \setminus C_{\gamma_{m-1}+1}^{(1)} &= \{(a, 0) : (-a)^{(q-1)/\gamma_{q,r,m-1}} = 1\}, \\ c_1 &= c_1^{(1)} + (q-1)(1 - \gamma_{q,r,m-1}^{-1}) + 1, \\ c_{\gamma_{m-1}+1} &= c_{\gamma_{m-1}+1}^{(1)} + (q-1)\gamma_{q,r,m-1}^{-1}. \end{aligned}$$

(ii) *We have $C_i^{(1)} = C_i^{(2)}$ for all $i \notin \{0, \gamma_m\}$, and*

$$\begin{aligned} C_0^{(1)} \setminus C_0^{(2)} &= \{(0, b) : (-b)^{(q-1)/\gamma_{q,r,m}} \neq 1\}, \\ C_{\gamma_m}^{(1)} \setminus C_{\gamma_m}^{(2)} &= \{(0, b) : (-b)^{(q-1)/\gamma_{q,r,m}} = 1\}, \\ c_0^{(1)} &= c_0^{(2)} + (q-1)(1 - \gamma_{q,r,m}^{-1}) \\ c_{\gamma_m}^{(1)} &= c_{\gamma_m}^{(2)} + (q-1)\gamma_{q,r,m}^{-1}. \end{aligned}$$

We note that Theorem 5.1 is also counting the number of possible solutions to the equations $x^{r+1} + ax + b$, as in Bluher's (2004) work. The comparison with Bluher's work is interesting because she does not consider the case $a = 0$ or $b = 0$ and because her work has multiple cases depending on whether d is even or odd and whether m is even or odd, whereas our counts have no such special cases.

The result in the (relatively straightforward) case $a = 0$ is consistent with the more general Lemma 5.9 of von zur Gathen (2008), where q is not required to be a power of r , but merely of p . As a corollary we obtain the counting result of Bluher (2004b) (at least over $\mathbb{F}_q$, when q is a power of r).

The constructive nature of our proofs allows us to build polynomials prescribed to be in any of these decomposition classes. This follows in the same manner as in the degree r^2 case. We generate elements of $\mathbb{F}_r[x; q]$ with the desired factorization pattern (which determines the number of collisions) and decompose these over $\mathbb{F}_q[x; r]$ using the algorithms of Giesbrecht (1998).

6. GENERAL COMPOSITIONS

The previous sections provide a good understanding of composition collisions for additive polynomials. We now move on to general polynomials of degree r^2 , and provide some explicit non-additive collisions.

For any $f = \sum f_i x^i \in \mathbb{F}_q[x]$, we call $\deg_2 f = \deg(f - \text{lc}(f)x^{\deg f})$ the *second-degree* of f , with $\deg_2 f = -\infty$ for monomials and zero.

THEOREM 6.1. Let q and r be powers of p , $\varepsilon \in \{0, 1\}$, $u, s \in \mathbb{F}_q^\times$, $t \in T = \{t \in \mathbb{F}_q : t^{r+1} - \varepsilon ut + u = 0\}$, ℓ a positive divisor of $r - 1$, $m = (r - 1)/\ell$, and

$$\begin{aligned} f &= F(\varepsilon, u, \ell, s) = x(x^{\ell(r+1)} - \varepsilon u s^r x^\ell + u s^{r+1})^m, \\ g &= G(u, \ell, s, t) = x(x^\ell - u s^r t^{-1})^m, \\ h &= H(\ell, s, t) = x(x^\ell - s t)^m, \end{aligned}$$

all in $\mathbb{F}_q[x]$. Then

$$f = g \circ h,$$

and f has a $\#T$ -collision.

If a polynomial $f \in \mathbb{F}_q[x]$ is monic original, then so is $f_{(w)} = (x - f(w)) \circ f \circ (x + w)$ for all $w \in \mathbb{F}_q$. Every decomposition of f induces a decomposition of $f_{(w)}$, and all $f_{(w)}$ have the same number of decompositions as $f_{(0)} = f$.

Among all $F(\varepsilon, u, \ell, s)_{(w)}$, the $F(\varepsilon, u, \ell, s)_{(0)}$ is characterized by the vanishing of the coefficient of $x^{r^2 - \ell r - \ell - 1}$.

PROPOSITION 6.2. Let q and r be powers of p . Let $\varepsilon, u, \ell, s, t$ and $\varepsilon^*, u^*, \ell^*, s^*, t^*$ satisfy the conditions of Theorem 6.1, $w, w^* \in \mathbb{F}_q$, $f = F(\varepsilon, u, \ell, s)_{(w)}$, and $f^* = F(\varepsilon^*, u^*, \ell^*, s^*)_{(w^*)}$. The following holds:

- (i) If $f = f^*$, then $\varepsilon = \varepsilon^*$ and $\ell = \ell^*$.
- (ii) If $\varepsilon = 1$ and $\ell < r - 1$, then $f = f^*$ if and only if $u = u^*$, $s = s^*$ and $w = w^*$.
- (iii) If $\varepsilon = 1$ and $\ell = r - 1$, then $f = F(1, u, r - 1, s)_{(0)}$ and $f = f^*$ if and only if $u = u^*$ and $s = s^*$.
- (iv) If $\varepsilon = 0$ and $\ell < r - 1$, then $f = F(0, -1, \ell, st)_{(w)}$ and $f = f^*$ if and only if $w = w^*$ and $(s/s^*)^{r+1} = 1$.
- (v) If $\varepsilon = 0$ and $\ell = r - 1$, then $f = F(0, -1, r - 1, st)_{(0)}$ and $f = f^*$ if and only if $(s/s^*)^{r+1} = 1$.

COROLLARY 6.3. Let p, q, r be as in Theorem 6.1, $\gamma = \gcd(r + 1, q - 1)$, $i \in \{2, r + 1\}$, and N_i the number of $F(\varepsilon, u, \ell, s)_{(w)}$ which have a maximal i -collision as constructed above. Then

$$N_i = (1 - q + q \cdot d(r - 1)) \left(c_{q,r,i}^{(2)} + \delta_{\gamma,i} \frac{q - 1}{\gamma} \right),$$

where $d(r - 1)$ is the number of divisors of $r - 1$, $\delta_{i,j}$ is Kronecker's delta, and $c_{q,r,i}^{(2)}$ are determined by Theorem 5.1 and Proposition 5.4.

Von zur Gathen (2008), Lemma 3.29, determines $\gcd(r + 1, q - 1)$ explicitly.

CONJECTURE 6.4. Any squarefree maximal i -collision with $i \geq 2$ at degree p^2 is of the form $\{(G(u, \ell, s, t)_{(w)}, H(\ell, s, t)_{(w)}) : t \in T\}$.

In the following, we present partial results on this conjecture, concentrating on the simplest case $r = p$. We also give an upper bound on the number of decompositions a single polynomial can have in the case of degree p^2 . No nontrivial estimate seems to be in the literature.

PROPOSITION 6.5. Let C be a non-Frobenius i -collision over $\mathbb{F}_q$ with $i \geq 2$ at degree p^2 . There is an integer k with $1 \leq k < p$ and the following properties for all $(g, h) \in C$.

$$(i) \deg_2(g) = \deg_2(h) = k.$$

(ii) For all $(g^*, h^*) \in C$ with $(g, h) \neq (g^*, h^*)$, we have $g_k \neq g_k^*$ and $h_k \neq h_k^*$.

(iii) Set $a = -f_{kp}$ and $b = k^{-1}f_{kp-p+k}$. Then $bh_k \neq 0$, and

$$\begin{aligned} h_k^{p+1} + ah_k + b &= 0 \\ g_k &= -a - h_k^p = bh_k^{-1}. \end{aligned}$$

(iv) $i \leq p + 1$.

We have $k = 1$ for additive polynomials, and $k = r - \ell$ in Theorem 6.1.

PROPOSITION 6.6. Take a non-Frobenius i -collision over $\mathbb{F}_q$ with $i \geq 2$ at degree p^2 , and let k be the integer defined in Proposition 6.5. Then $k = 1$ or $k > p/2$. In particular, there are no collisions at degree p^2 with $k = 2$ if $p > 3$ nor with $k = 3$ if $p > 5$.

7. CONCLUSION AND OPEN QUESTIONS

We have presented composition collisions with component degrees (r, r) for polynomials f of degree r^2 , and observed a fascinating interplay between these examples—quite distinct in the additive and the $f_{r^2-r-1} \neq 0$ cases—and Abhyankar's projective polynomials and Blüher's statistics on their roots. Furthermore, we showed that our examples comprise all possibilities in the additive case, and provided large classes of examples in general. Showing the completeness of our examples in the general case is the main challenge left open here as Conjecture 6.4.

Generalizations go in two directions. One is degree r^k for $k \geq 3$. Additive polynomials are of special interest here, and the rational normal form of the Frobenius automorphism will play a major role. For general polynomials, the approximate counting problem is solved in von zur Gathen (2009) with a relative error of about q^{-1} , and it is desirable to reduce this, say to q^{-r+1} . The second direction is to look at degree ar^2 with $r \nmid a$. Now there are no additive polynomials, but for approximate counting, the best known relative error can be as large as 1. It would be interesting to also push this below q^{-1} , or even q^{-r+1} .

In some sections, we assume the field size q to be a power of the parameter r . As in Blüher's (2004) work, our methods go through for the general situation, where q and r are independent powers of the characteristic.

With respect to additive polynomials, a more thorough computational investigation of projective polynomials is warranted. Automatic generation of Blüher-like counting results for higher degree projective polynomials should be possible, as would be a more exact understanding of their possible collision numbers.

8. ACKNOWLEDGMENTS

The authors thank Toni Blüher for telling us about the applications of projective polynomials, and an anonymous referee for pointing us to Hellesteth & Kholosha (2010).

The work of Joachim von zur Gathen and Konstantin Ziegler was supported by the B-IT Foundation and the Land Nordrhein-Westfalen. The work of Mark Giesbrecht was supported by NSERC Canada and MITACS.

References

- SHREERAM S. ABHYANKAR. Projective Polynomials. *Proceedings of the American Mathematical Society*, **125**(6):1643–1650, 1997. ISSN 00029939. URL <http://www.jstor.org/stable/2162203>.
- LEONARD M. ADLEMAN & HENDRIK W. LENSTRA, JR. Finding Irreducible Polynomials over Finite Fields. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, Berkeley CA*, pages 350–355. ACM Press, 1986.
- DAVID R. BARTON & RICHARD ZIPPEL. Polynomial Decomposition Algorithms. *Journal of Symbolic Computation*, **1**:159–168, 1985.
- ANTONIA W. BLUHER. On $x^6 + x + a$ in Characteristic Three. *Designs, Codes and Cryptography*, **30**:85–95, 2003. URL <http://www.springerlink.com/content/r213567443r63360/fulltext.pdf>.
- ANTONIA W. BLUHER. Explicit formulas for strong Davenport pairs. *Acta Arithmetica*, **112**(4):397–403, 2004a.
- ANTONIA W. BLUHER. On $x^{q+1} + ax + b$. *Finite Fields and Their Applications*, **10**(3):285–305, 2004b. URL <http://dx.doi.org/10.1016/j.ffa.2003.08.004>.
- ANTONIA W. BLUHER & ALAIN LASJAUNIAS. Hyperquadratic power series of degree four. *Acta Arithmetica*, **124**(3):257–268, 2006.
- JOHN J. CADE. A New Public-key Cipher Which Allows Signatures. In *Proceedings of the 2nd SIAM Conference on Applied Linear Algebra*, Raleigh NC A11. SIAM, 1985.
- J. F. DILLON. Geometry, codes and difference sets: exceptional connections. In *Codes and designs (Columbus, OH, 2000)*, volume 10 of *Ohio State Univ. Math. Res. Inst. Publ.*, pages 73–85. de Gruyter, Berlin, 2002. URL <http://dx.doi.org/10.1515/9783110198119.73>.
- F. DOREY & G. WHAPLES. Prime and Composite Polynomials. *Journal of Algebra*, **28**:88–101, 1974. URL [http://dx.doi.org/10.1016/0021-8693\(74\)90023-4](http://dx.doi.org/10.1016/0021-8693(74)90023-4).
- JOACHIM VON ZUR GATHEN. Functional Decomposition of Polynomials: the Tame Case. *Journal of Symbolic Computation*, **9**:281–299, 1990a. URL [http://dx.doi.org/10.1016/S0747-7171\(08\)80014-4](http://dx.doi.org/10.1016/S0747-7171(08)80014-4).
- JOACHIM VON ZUR GATHEN. Functional Decomposition of Polynomials: the Wild Case. *Journal of Symbolic Computation*, **10**:437–452, 1990b. URL [http://dx.doi.org/10.1016/S0747-7171\(08\)80054-5](http://dx.doi.org/10.1016/S0747-7171(08)80054-5).
- JOACHIM VON ZUR GATHEN. Counting decomposable univariate polynomials. *Preprint*, 92 pages, 2008. URL <http://arxiv.org/abs/0901.0054>.
- JOACHIM VON ZUR GATHEN. An algorithm for decomposing univariate wild polynomials. *Journal of Symbolic Computation*, to appear, 32 pages, 2010.
- JOACHIM VON ZUR GATHEN. The Number of Decomposable Univariate Polynomials. In JOHN P. MAY, editor, *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation ISSAC2009, Seoul, Korea*, pages 359–366. 2009. ISBN 978-1-60558-609-0.
- JOACHIM VON ZUR GATHEN, MARK GIESBRECHT & KONSTANTIN ZIEGLER. Composition collisions and projective polynomials, 2010. URL <http://arxiv.org/abs/1005.1087>.
- MARK WILLIAM GIESBRECHT. Complexity Results on the Functional Decomposition of Polynomials. Technical Report 209/88, University of Toronto, Department of Computer Science, Toronto, Ontario, Canada, 1988. Available as <http://arxiv.org/abs/1004.5433>.
- MARK GIESBRECHT. Nearly Optimal Algorithms for Canonical Matrix Forms. *SIAM J. Comp.*, **24**:948–969, 1995.
- MARK GIESBRECHT. Factoring in Skew-Polynomial Rings over Finite Fields. *Journal of Symbolic Computation*, **26**(4):463–486, 1998. URL <http://dx.doi.org/10.1006/jSCO.1998.0224>.
- G. H. HARDY & S. RAMANUJAN. Asymptotic formulae in combinatory analysis. *Proceedings of the London Mathematical Society*, **17**(2):75–115, 1918.
- TOR HELLESETH & ALEXANDER KHOLOSHA. $x^{2^l+1} + x + a$ and related affine polynomials over $GF(2^k)$. *Cryptography and Communications*, **2**(1):85–109, 2010.
- TOR HELLESETH, ALEXANDER KHOLOSHA & AINA JOHANSEN. m -Sequences of Different Lengths with Four-Valued Cross Correlation. *IEEE International Symposium on Information Theory*, 2008.
- DEXTER KOZEN & SUSAN LANDAU. Polynomial Decomposition Algorithms. *Journal of Symbolic Computation*, **7**:445–456, 1989. An earlier version was published as Technical Report 86-773, Department of Computer Science, Cornell University, Ithaca NY, 1986.
- S. LANDAU & G. L. MILLER. Solvability by Radicals is in Polynomial Time. *Journal of Computer and System Sciences*, **30**:179–208, 1985.
- RUDOLF LIDL & HARALD NIEDERREITER. *Finite Fields*. Number 20 in Encyclopedia of Mathematics and its Applications. Addison-Wesley, Reading MA, 1983.
- ROBERT WINSTON KEITH ODOI. On additive polynomials over a finite field. *Proceedings of the Edinburgh Mathematical Society*, **42**:1–16, 1999.
- O. ORE. On a Special Class of Polynomials. *Transactions of the American Mathematical Society*, **35**:559–584, 1933.
- ANDRZEJ SCHINZEL. *Selected Topics on Polynomials*. Ann Arbor; The University of Michigan Press, 1982. ISBN 0-472-08026-1.
- ANDRZEJ SCHINZEL. *Polynomials with special regard to reducibility*. Cambridge University Press, Cambridge, UK, 2000. ISBN 0521662257.
- XIANGYONG ZENG, NIAN LI & LEI HU. A class of nonbinary codes and their weight distribution. *ArXiv e-prints, arxiv 0802.3430v1*, 2008. URL http://arxiv.org/PS_cache/arxiv/pdf/0802/0802.3430v1.pdf.
- RICHARD ZIPPEL. Rational Function Decomposition. In STEPHEN M. WATT, editor, *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation ISSAC '91, Bonn, Germany*, pages 1–6. ACM Press, Bonn, Germany, 1991. ISBN 0-89791-437-6.

Decomposition of Generic Multivariate Polynomials

Jean-Charles Faugère

SALSA Project INRIA Paris-Rocquencourt
UPMC, Univ Paris 06, LIP6
CNRS, UMR 7606, LIP6
UFR Ingénierie 919, LIP6 Passy Kennedy
4, Place Jussieu 75252 Paris Cedex 05
Jean-Charles.Faugere@inria.fr

Joachim von zur Gathen

B-IT, Universität Bonn
D-53113 Bonn, Germany
gathen@bit.uni-bonn.de

Ludovic Perret

SALSA Project INRIA Paris-Rocquencourt
UPMC, Univ Paris 06, LIP6
CNRS, UMR 7606, LIP6
UFR Ingénierie 919, LIP6 Passy Kennedy
4, Place Jussieu 75252 Paris Cedex 05
ludovic.perret@lip6.fr

ABSTRACT

We consider the composition $f = g \circ h$ of two systems $g = (g_0, \dots, g_t)$ and $h = (h_0, \dots, h_s)$ of homogeneous multivariate polynomials over a field $\mathbb{K}$, where each $g_j \in \mathbb{K}[y_0, \dots, y_s]$ has degree ℓ , each $h_k \in \mathbb{K}[x_0, \dots, x_r]$ has degree m , and $f_i = g_i(h_0, \dots, h_s) \in \mathbb{K}[x_0, \dots, x_r]$ has degree $n = \ell \cdot m$, for $0 \leq i \leq t$. The motivation of this paper is to investigate the behavior of the decomposition algorithm **MultiComPoly** proposed at ISSAC'09 [18]. We prove that the algorithm works correctly for generic decomposable instances – in the special cases where ℓ is 2 or 3, and m is 2 – and investigate the issue of uniqueness of a *generic* decomposable instance. The uniqueness is defined w.r.t. the “normal form” of a multivariate decomposition, a new notion introduced in this paper, which is of independent interest.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms Algebraic Algorithms

General Terms

Algorithms

Keywords

Functional Decomposition, Generic Uniqueness, Gröbner bases

1. INTRODUCTION

Let $\mathbb{K}$ be an arbitrary field. The multivariate *Functional Decomposition Problem (FDP)* [23, 12, 30] is the problem of representing a given polynomial $f = (f_0, \dots, f_t) \in \mathbb{K}[x_0, \dots, x_r]^{t+1}$ as a functional composition:

$$(f_0, \dots, f_t) = (g_0(h_0, \dots, h_s), \dots, g_t(h_0, \dots, h_s)),$$

of polynomials $g = (g_0, \dots, g_t) \in \mathbb{K}[y_0, \dots, y_s]^{t+1}$ and $h = (h_0, \dots, h_s) \in \mathbb{K}[x_0, \dots, x_r]^{s+1}$ of smaller degree.

FDP is a classical problem in computer algebra ([26, 21, 22, 23, 10, 29]) which has been thoroughly investigated in the univariate case

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

from an algorithmic as well as from a theoretical point of view; see [1, 5, 26, 21, 22, 20, 14, 27]. The decomposition of univariate polynomials is a standard functionality proposed by major computer algebra systems¹.

For general multivariate decomposition, the situation is different and probably more complicated. For instance, there is no multivariate equivalent of Ritt's theorem [27, 14] which is a central tool in the univariate case. Typically, this makes it delicate to define a proper notion of nontrivial decomposition (for instance see [23, 24]). In [23], von zur Gathen, Gutierrez and Rubio have investigated several variants of FDP, the so-called *uni-multivariate*, *multi-univariate* and *single-variable* decompositions, which are extensions of the univariate case. They presented algorithms to solve these variants, together with some theoretical results. It is only recently that algorithms for decomposing general multivariate polynomials have been proposed [17, 18]. The original motivation of these methods was in the cryptanalysis of multivariate cryptosystems [16]. In this paper, we focus attention on the **MultiComPoly** algorithm proposed at ISSAC'09 [18]. We are interested in the behavior of the algorithm for generic decomposable instances, in the special cases where ℓ is 2 or 3, and m is 2. These are sufficient for the cryptanalytic applications. We prove that the algorithm works correctly for generic decomposable instances, and returns a unique decomposition. The uniqueness is defined w.r.t. the “normal form” of a multivariate decomposition, a new notion introduced in this paper.

1.1 The MultiComPoly algorithm

In order to be self-contained, we briefly recall the principle of the decomposition algorithm **MultiComPoly** [18]. Some of the notation will be used in the rest of this paper. So, let $f = g \circ h$ be the composition of $g = (g_0, \dots, g_t) \in \mathbb{K}[y_0, \dots, y_s]^{t+1}$ and $h = (h_0, \dots, h_s) \in \mathbb{K}[x_0, \dots, x_r]^{s+1}$ of homogeneous multivariate polynomials. Most decomposition techniques first determine the right component h , then the left component g . The algorithm of [18] is no exception. More precisely, **MultiComPoly** recovers first the vector space $\mathcal{L}(h) = \text{Span}_{\mathbb{K}}(h_0, \dots, h_s)$ spanned by the right component h . This vector space is obtained by considering the ideal generated by high order differentials of f :

$$\partial^k \mathcal{J}_f = \left\langle \frac{\partial^k f_i}{\partial x_{j_1} \dots \partial x_{j_k}} \mid 0 \leq i \leq t, 0 \leq j_1 < \dots < j_k \leq r \right\rangle,$$

for some k depending of the degree of g , where $\mathcal{J}_f$ is the ideal generated by the polynomials in f . It has been proved [18] that there exists $\delta > 0$ such that:

$$x_i^\delta h_i \subseteq \partial^{\deg(g)-1} \mathcal{J}_f, \text{ for all } i, 0 \leq i \leq s.$$

¹For instance, compoly of MAPLE <http://www.maplesoft.com/>

A basis of $\mathcal{L}(h)$ is obtained by computing a DRL (degree reverse lexicographical) Gröbner basis [6, 7, 8, 9] of $\partial^{\deg(g)-1} \mathcal{J}_f : x^\delta$, for a suitable $\delta > 0$. More precisely, we compute a truncated [11] Gröbner basis G of $\partial^{\deg(g)-1} \mathcal{J}_f : x^\delta$. If $\#G = s+1$, then $\text{Span}_{\mathbb{K}}(G) = \mathcal{L}(h)$. From the knowledge of $\mathcal{L}(h)$, it is well known [28] that the left component g can be recovered by solving a linear system of equations. This is studied in more generality in Section 4.

1.2 Organization of the paper

We study in detail the behavior of **MultiComPoly** for generic decomposable instances. The paper is organized as follows. In Section 2, we introduce more precisely the decomposition problem studied here, and fix some further notation. In Section 3, we focus on the first part of **MultiComPoly** which computes the vector space $\mathcal{L}(h)$. Let the notation be as in subsection 1.1, and G be the set of polynomials computed during the first step of **MultiComPoly** in Section 3, we prove that the property:

$$\text{Span}_{\mathbb{K}}(G) = \mathcal{L}(h).$$

is generic (in the sense of the Zariski topology). We first prove that the set of elements for which this property fails is contained in a closed algebraic set. The second part of the proof, which is the most difficult, consists of finding particular decomposable instances for which we can prove the property. As a side remark, we mention that the genericity of semi-regular sequences [2, 3, 4] is a well known conjecture of Fröberg [19] whose bottleneck is to simply find a semi-regular sequence. In our context, we consider in Section 3 rather simple family of decomposable instances. For this family, we prove that the equality $\text{Span}_{\mathbb{K}}(G) = \mathcal{L}(h)$ indeed holds. To do that, we describe the exact structure of the truncated Gröbner basis G for the family under consideration. After that, we study in Section 4 the property of the linear system corresponding to the recovery of the left component when the right component is known. We conjecture that for a “generic” h , the system has maximal rank and thus is overdetermined. This conjecture has been proven in the previous sections for the examples considered there.

All in all, we prove that **MultiComPoly** computes a “unique” decomposition, w.r.t a normal form, for generic decomposable instances.

2. FUNCTIONAL DECOMPOSITION

Rather than the general multivariate *Functional Decomposition Problem* (FDP) problem (see [23, 12, 30]), we consider throughout this paper the homogeneous variant. Thus for any positive integers ℓ and m , we have the following problem.

FDP(ℓ, m)

Input: $f = (f_0, \dots, f_t) \in \mathbb{K}[x_0, \dots, x_r]^{t+1}$ homogeneous polynomials, all of the same degree.

Output: Either “no decomposition” or homogeneous polynomials $(g = (g_0, \dots, g_t), h = (h_0, \dots, h_s)) \in \mathbb{K}[y_0, \dots, y_s]^{t+1} \times \mathbb{K}[x_0, \dots, x_r]^{s+1}$ all of degree ℓ and m , respectively, such that $h = f \circ g$.

Trivial decomposition may occur when $\ell = 1$ or $m = 1$, and we assume in the rest of this paper that $\ell > 1$ and $m > 1$.

DEFINITION 1. $f \in \mathbb{K}[x_0, \dots, x_r]^{t+1}$ is decomposable if there exists (g, h) such that $f = g \circ h$ with $\deg(g) > 1$ and $\deg(h) > 1$. The pair (g, h) is an (ℓ, m) decomposition of f if (g, h) is a decomposition of f with $\deg(g) = \ell$ and $\deg(h) = m$.

Linear substitutions introduce inessential nonuniquenesses of decompositions. Indeed, any invertible linear combination $A \in \text{GL}_s(\mathbb{K})$

of $(h_0, \dots, h_s)$ leads to a decomposition of f , since

$$f(x_1, \dots, x_r) = (g(y_0, \dots, y_r)A^{-1}) \circ (h(x_0, \dots, x_r)A).$$

As in the univariate case, it is convenient to define a “normal form” [21, 22, 20] of a decomposition. In the univariate case, a polynomial h is said to be original if $h(0) \neq 0$. A univariate decomposition (g, h) of f is called normal if h is original and monic (i.e., leading coefficient equal to 1). We introduce a similar notion for the multivariate case.

DEFINITION 2. We consider homogeneous monic polynomials, whose leading coefficient in the DRL order equals 1. A decomposition (g, h) of such an f is in normal form if the polynomials $((g_0, \dots, g_t), (h_0, \dots, h_s))$ are homogeneous and monic and $(h_0, \dots, h_s)$ is an m -Gröbner basis (a Gröbner basis up to degree m) w.r.t. DRL order (i.e., degree reverse lexicographical). Two decompositions (g, h) and $(\tilde{g}, \tilde{h})$ of f are equivalent if their normal forms are equal.

In the multivariate case, the fact that $(h_0, \dots, h_s)$ are homogeneous implies in particular $h(\mathbf{0}) = 0$. One might view homogeneous as a natural extension of the concept of original. In addition, if the polynomials of h are an m -Gröbner basis, then the polynomials $(h_0, \dots, h_s)$ are, in particular, monic. Note that if h is a m -Gröbner basis, then $(h_0, \dots, h_s)$ is also a basis of the $\mathbb{K}$ -vector space spanned by $h_0, \dots, h_s$; a natural and canonical representative of equivalent decompositions. Note that **MultiComPoly** actually computes the normal form of a decomposition.

We fix some notation for the remainder of this paper. For $r \geq 1$ and $\delta \geq 0$, we write:

$$P_{r, \delta} = \{f \in \mathbb{K}[x_0, \dots, x_r] : f \text{ homogeneous, and } \deg(f) = \delta\}$$

for the vector space of homogeneous polynomials of degree δ . A basis of $P_{r, \delta}$, denoted $M_r(\delta)$, is given by the set of all monomials of degree δ . Thus $\dim(P_{r, \delta}) = \#M_r(\delta)$. We define the composition map:

$$\gamma_{s, \ell, r, m} : \begin{array}{ccc} P_{s, \ell} \times P_{r, m} & \rightarrow & P_{r, \ell, m} \\ (g, h) & \mapsto & g \circ h \end{array}$$

and write $D_{r, \ell, m} = \text{Im}(\gamma_{s, \ell, r, m})$ for the set of (ℓ, m) decomposables. Finally, we state the framework in which we prove our results.

DEFINITION 3. Let F be an algebraic closure of $\mathbb{K}$, and $E_{\ell, m} \subset F[y_0, \dots, y_s]^{t+1} \times F[x_0, \dots, x_r]^{s+1}$ be the set of homogeneous polynomials $(g_0, \dots, g_t)$ of degree ℓ , and $(h_0, \dots, h_s)$ of degree m . We say that a property is generic if the set of elements in $E_{\ell, m}$ verifying this property is a non-empty Zariski-open subset; i.e., the property is verified for all elements of $E_{\ell, m}$ except for an algebraic set of codimension one.

We recall that in order to prove that a certain property is generic, it is sufficient to show the following

1. First: show that the set of points/elements for which the property fails is the zero of a system of polynomial equations. This defines the complement of an open set with respect to Zariski topology.
2. Second: prove that the Zariski-open subset is not empty; which means that we have to prove that the property is valid at least on one specific example. The examples that we exhibit are actually defined over the ground field $\mathbb{K}$, and we avoid reference to its algebraic closure in the following.

3. GENERIC UNIQUENESS OF THE RIGHT COMPONENT

We consider here the first part of **MultiComPoly** on the set $D_{r,\ell,m}$ of (ℓ, m) decomposables. The aim of the first part is to obtain a basis of the vector space $\mathcal{L}(h)$. As explained in the introduction, this vector space is obtained from the truncated m -Gröbner basis G of $\partial^{\ell-1} \mathcal{J}_f : x_r^\delta$, for a suitable $\delta > 0$, w.r.t. DRL. In [18], it is proved that $\text{Span}_{\mathbb{K}}(G)$ is also a basis of $\mathcal{L}(h)$ as a $\mathbb{K}$ -vector space, if $\#G = s + 1$. We prove here that the property

$$\text{Span}_{\mathbb{K}}(G) = \mathcal{L}(h)$$

is generic for the set of $D_{r,2,2}$ of $(2, 2)$ decomposables, and for the set of $D_{r,3,2}$ of $(3, 2)$ decomposables.

3.1 Roadmap of the proof

In both cases $D_{r,2,2}$ and $D_{r,3,2}$, the general strategy is identical although the technical details differ. As explained previously, a proof of genericity is divided into two steps. We provide here a high level description of the strategy in our context.

1. To define the algebraic set, we will adopt a linear algebra point of view. In this context, it is not difficult to see that the condition $\mathcal{L}(h) \neq \text{Span}_{\mathbb{K}}(G)$ implies a defects in the rank of a certain matrix. By considering generic polynomials, it is possible to construct an algebraic system whose variables correspond to the coefficients of a right component. This algebraic system vanishes as soon as the right component h is such that $\mathcal{L}(h) \neq \text{Span}_{\mathbb{K}}(G)$.
2. We prove then that the Zariski-open set is not empty by providing suitable explicit examples. This is the most difficult part of the proof. Here, we will use a polynomial point of view. We consider the following family $f = g \circ h \in D_{r,\ell,2}$ of $(\ell, 2)$ decomposables:

- $r = s = t$ and $g = (y_0^\ell, \dots, y_s^\ell)$,
- for all i with $0 \leq i \leq s$, $h_i = \sum_{j=i}^s x_j^2$.

3.2 $(2, 2)$ decomposition

We first consider the basic case of a decomposable $f \in D_{r,2,2}$. Let then $((g_0, \dots, g_t), (h_0, \dots, h_s))$ be a $(2, 2)$ decomposition of f . In this situation, we have to consider the ideal:

$$\partial \mathcal{J}_f = \left\langle \frac{\partial f_i}{\partial x_u} \mid 0 \leq i \leq t, \text{ and } 0 \leq u \leq r \right\rangle.$$

generated by the partial derivatives of f . This is due to the fact that for all $0, 1 \leq i \leq t$, $f_i = g_i(h_0, \dots, h_s) = \sum_{0 \leq j, k \leq r} g_{j,k}^{(i)} h_j h_k$, with $g_i = \sum_{0 \leq j, k \leq s} g_{j,k}^{(i)} y_j y_k$. Thus

$$\frac{\partial f_i}{\partial x_u} = \sum_{0 \leq j, k \leq s} g_{j,k}^{(i)} \left(h_j \frac{\partial h_k}{\partial x_u} + h_k \frac{\partial h_j}{\partial x_u} \right).$$

Each partial derivative $\frac{\partial f_i}{\partial x_u}$ is a linear combination of elements $\{x_j \cdot h_k\}_{0 \leq k \leq s}^{0 \leq j \leq r}$. For the analysis, it is convenient to consider the $((t+1) \cdot$

$(r+1)) \times ((s+1) \cdot (r+1))$ matrix:

$$A = \begin{matrix} & \dots & \dots & x_j \cdot h_k & \dots & \dots \\ \frac{\partial f_0}{\partial x_u} & \left(\begin{array}{cccccc} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{array} \right) & \dots & \dots & \dots & \dots & \dots \\ \vdots & & & & & & & & & & & \\ \frac{\partial f_i}{\partial x_u} & & & & & & & & & & & \\ \vdots & & & & & & & & & & & \\ \frac{\partial f_s}{\partial x_u} & & & & & & & & & & & \end{matrix} \quad (1)$$

where the $((i, u), (j, k))$ -entry equals the coefficient of $x_j \cdot h_k$ in $\frac{\partial f_i}{\partial x_u}$. If $\text{Rank}(A) = \# \text{Columns}(A) = (s+1) \cdot (r+1)$, then each $x_j \cdot h_k$ can be expressed as a linear combination of $\frac{\partial f_i}{\partial x_u}$ leading in particular to

$$x_r h_i \in \partial \mathcal{J}_f, \text{ for all } i, 0 \leq i \leq s. \quad (2)$$

Let G be a truncated 2-Gröbner basis of $\partial \mathcal{J}_f : x_r$. Our goal is to prove that

$$\text{Span}_{\mathbb{K}}(G) = \mathcal{L}(h). \quad (3)$$

This condition (3) is clearly a necessary condition of success of **MultiComPoly**. The set of decomposable for which (3) is not fulfilled is an algebraic set. Indeed, the failure of condition (3) is due to a defect in the rank of two sub matrices of (1) (see [18]). It remains to prove that this Zariski-open set is nonempty. To do so, we consider the following particular decomposable instance $f = g \circ h \in D_{r,2,2}$:

- $r = s = t$ and $g = (y_0^2, \dots, y_s^2)$
- for all $i, 0 \leq i \leq s$, $h_i = \sum_{j=i}^s x_j^2$.

To show that (3) is fulfilled for this family, we need several intermediate results.

LEMMA 3.1. *Let $f = g \circ h \in D_{r,2,2}$ be as defined previously. For all $i, 0 \leq i \leq s$, we have:*

$$\frac{\partial f_i}{\partial x_u} = \begin{cases} 4x_u h_i = 4x_u \sum_{j=i}^s x_j^2 & \text{if } u \geq i, \\ 0 & \text{if } u < i. \end{cases}$$

PROOF.

$$\begin{aligned} f_i &= h_i^2, \\ \frac{\partial f_i}{\partial x_u} &= 2h_i \frac{\partial h_i}{\partial x_u}. \end{aligned}$$

Due to the particular choice of h , $\frac{\partial f_i}{\partial x_u} = 0$ if $u < i$. For all $u \geq i$, $\frac{\partial f_i}{\partial x_u} = 4x_u h_i = 4x_u \sum_{j=i}^s x_j^2$. $\square$

From this, we deduce the following.

LEMMA 3.2. *For all $i \leq s$ and $u > i$:*

$$\frac{\partial f_i}{\partial x_u} - \frac{\partial f_{i+1}}{\partial x_u} = 4x_u x_i^2,$$

with the convention that $f_{s+1} = f_0$.

Recall that we consider the DRL ordering $\succ$ with $x_0 \succ \dots \succ x_s$.

LEMMA 3.3. *Let $i \leq s$. Then*

$$LT_{\succ} \left(\frac{\partial f_i}{\partial x_i} \right) = x_i^3,$$

where $LT_{\succ}$ stands for the leading term.

PROOF. Here, $\frac{\partial f_i}{\partial x_i} = 4x_i \sum_{j=i}^s x_j^2$. Hence:

$$LT_{>} \left(\frac{\partial f_i}{\partial x_i} \right) = x_i LT_{>} \left(\sum_{j=i}^s x_j^2 \right) = x_i^3.$$

□

We now describe explicitly the leading terms of $\partial \mathcal{J}_f$.

LEMMA 3.4. *Let $f = g \circ h \in D_{r,2,2}$ be the particular example defined previously. The leading terms of a truncated 3-Gröbner basis of $\partial \mathcal{J}_f$ are:*

$$\begin{aligned} & \left[x_s^3 \right] \cup \\ & \left[x_s x_{s-1}^2, x_{s-1}^3 \right] \cup \left[x_s x_{s-2}^2, x_{s-1} x_{s-2}^2, x_{s-2}^3 \right] \\ & \cup \dots \\ & \cup \left[x_s x_0^2, x_{s-1} x_0^2, \dots, x_2 x_0^2, x_0^3 \right]. \end{aligned}$$

PROOF. Clearly

$$\begin{aligned} \partial \mathcal{J}_f &= \left\langle \frac{\partial f_i}{\partial x_u} \mid 0 \leq i \leq u \leq s \right\rangle \\ &= \left\langle \frac{\partial f_i}{\partial x_u} \mid 0 \leq i < u \leq s \right\rangle + \left\langle \frac{\partial f_i}{\partial x_i} \mid 0 \leq i \leq s \right\rangle \\ &= \left\langle \frac{\partial f_i}{\partial x_u} - \frac{\partial f_{i+1}}{\partial x_u} \mid 0 \leq i < u \leq s \right\rangle + \left\langle \frac{\partial f_i}{\partial x_i} \mid 0 \leq i \leq s \right\rangle \\ &= \left\langle x_u x_i^2 \mid 0 \leq i < u \leq s \right\rangle + \left\langle \frac{\partial f_i}{\partial x_i} \mid 0 \leq i \leq s \right\rangle. \end{aligned}$$

Since $LT_{>} \left(\frac{\partial f_i}{\partial x_i} \right) = x_i^3$, the leading terms are pairwise distinct. This proves that

$$\left[x_u x_i^2 \mid 0 \leq i < u \leq s \right] + \left[\frac{\partial f_i}{\partial x_i} \mid 0 \leq i \leq s \right],$$

is a 3-Gröbner basis of $\partial \mathcal{J}_f$. □

Finally:

COROLLARY 3.1. *Let $\mathbb{K}$ be a field of characteristic $\neq 2$, and let $f = g \circ h \in D_{r,2,2}$ be the particular example defined previously. The truncated 2-Gröbner basis of $\partial \mathcal{J}_f : x_s$ is exactly $[x_0^2, \dots, x_s^2] = \mathcal{L}(h)$.*

PROOF. It is a well known property of the DRL ordering that for a polynomial f , $x_s \mid f$ iff $x_s \mid LT_{>}(f)$. Consequently, the polynomials in $\partial \mathcal{J}_f$ of degree 3 divisible by x_s are, thanks to Lemma 3.4: $\frac{\partial f_0}{\partial x_s} = 4x_s \sum_{j=0}^s x_j^2$ and $x_s x_i^2$ for $0 \leq i < s$. Consequently, the truncated 2-Gröbner basis of $\partial \mathcal{J}_f : x_s$ is:

$$\left\langle \sum_{j=0}^s x_j^2, x_0^2, \dots, x_{s-1}^2 \right\rangle = \langle x_s^2, x_0^2, \dots, x_{s-1}^2 \rangle.$$

Finally, is not difficult to see that a basis of $\mathcal{L}(h)$ is also $[x_0^2, \dots, x_s^2]$. □

3.3 (3,2) decomposition

We now consider a (3,2) decomposable $f = (f_0, \dots, f_t) \in D_{r,2,3}$. In this case, we start from the ideal generated by the second order partial derivatives:

$$\partial^2 \mathcal{J}_f = \left\langle \frac{\partial^2 f_i}{\partial x_u \partial x_p} \mid 0 \leq i \leq t, \text{ and } 0 \leq u, p \leq r \right\rangle.$$

According to [18], each generator of the previous ideal is a linear combination of elements $\{x_j x_k \cdot h_q\}_{1 \leq j,k \leq r}^{1 \leq q \leq s}$. As previously, it is convenient to consider the $(t \cdot r(r+1)/2) \times (s \cdot r(r+1)/2)$ matrix:

$$A = \frac{\partial^2 f_0}{\partial x_u \partial x_p} \begin{pmatrix} \dots & \dots & x_j x_k \cdot h_q & \dots & \dots \\ \vdots & & \vdots & & \vdots \\ \frac{\partial^2 f_i}{\partial x_u \partial x_p} & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ \frac{\partial^2 f_t}{\partial x_u \partial x_p} & & \vdots & & \vdots \end{pmatrix}$$

In a similar way, if $\text{Rank}(A) = \# \text{Columns}(A)$, then each $x_r^2 \cdot h_i$ can be expressed as a linear combination of $\frac{\partial^2 f_i}{\partial x_u \partial x_p}$ leading in particular to

$$x_r^2 h_i \in \partial \mathcal{J}_f, \text{ for all } i, 0 \leq i \leq s. \quad (4)$$

Let G be truncated 2-Gröbner basis of $\partial \mathcal{J}_f : x_r^2$. Again, we want to prove the necessary condition of success of **MultiCompPoly**:

$$\text{Span}_{\mathbb{K}}(G) = \mathcal{L}(h). \quad (5)$$

Similarly to the (2,2) case, it is clear that set of h satisfying (5) is a Zariski-open set. The main task is to show that it is nonempty. We consider the same type of decomposable $f = g \circ h \in D_{r,3,2}$ as previously:

- $r = s = t$ and $g = (y_0^3, \dots, y_s^3)$,
- for all $i, 0 \leq i \leq s, h_i = \sum_{j=i}^s x_j^2$.

In what follows, we set $f_{s+1} = h_{s+1} = 0$. The idea is to split the ideal $\partial^2 \mathcal{J}_f$ into several parts:

$$\partial^2 \mathcal{J}_f = \mathcal{H}_1 \cap \mathcal{H}_2 \cap \mathcal{H}_3$$

where:

$$\begin{aligned} \mathcal{H}_1 &= \left\langle \frac{\partial^2 f_{i+1}}{\partial x_u \partial x_p} - \frac{\partial^2 f_i}{\partial x_u \partial x_p} \mid i < u < p \leq s \right\rangle \\ \mathcal{H}_2 &= \left\langle \frac{\partial^2 f_{i+1}}{\partial x_u^2} - \frac{\partial^2 f_i}{\partial x_u^2} \mid 0 \leq i < s \text{ and } i \leq u \leq s \right\rangle + \left\langle \frac{\partial^2 f_s}{\partial x_s^2} \right\rangle \\ \mathcal{H}_3 &= \left\langle \frac{\partial^2 f_{i+1}}{\partial x_i \partial x_p} - \frac{\partial^2 f_i}{\partial x_i \partial x_p} \mid 0 \leq i < s \text{ and } p \leq s \right\rangle \end{aligned}$$

It turns out that we can predict accurately the leading terms of a 4-Gröbner basis of each ideals and that they are all distinct. For that, we need several technical lemmas.

LEMMA 3.5. *For $i < u < p \leq s$, we have:*

$$\begin{aligned} \frac{\partial^2 f_i}{\partial x_u \partial x_p} &= 24x_u x_p h_i \\ \frac{\partial^2 f_{i+1}}{\partial x_u \partial x_p} - \frac{\partial^2 f_i}{\partial x_u \partial x_p} &= 24x_i^2 x_u x_p. \\ \frac{\partial^2 f_i}{\partial x_u^2} &= 6(h_i + 4x_u) h_i. \end{aligned} \quad (6)$$

Consequently $\mathcal{H}_1 = \langle x_i^2 x_u x_p \mid i < u < p \leq s \rangle$.

PROOF.

$$\begin{aligned} f_i &= h_i^3, \\ \frac{\partial f_i}{\partial x_u} &= 3h_i^2 \frac{\partial h_i}{\partial x_u}. \end{aligned}$$

Due to the particular choice of h , $\frac{\partial f_i}{\partial x_u} = 0$ if $u < i$ and for all $o \geq$

$i, \frac{\partial f_i}{\partial x_u} = 6x_u h_i^2$. Now, let $i \leq u \leq p \leq s$, we have $\frac{\partial^2 f_i}{\partial x_u \partial x_p} =$

$$\begin{aligned} 12x_u h_i \frac{\partial h_i}{\partial x_p} &= 24x_u x_p h_i, & \text{if } u \neq p. \\ &= 6h_i^2 + 24x_u^2 h_i, & \text{if } u = p. \end{aligned}$$

Finally, if $u \neq p$, then $\frac{\partial^2 f_{i+1}}{\partial x_u \partial x_p} - \frac{\partial^2 f_i}{\partial x_u \partial x_p} = x_u x_p (h_{i+1} - h_i) = x_u x_p x_i^2$. $\square$

LEMMA 3.6. *The leading terms w.r.t a DRL ordering of a truncated 4-Gröbner basis of $\mathcal{H}_3$ have the following shape:*

$$x_i^3 x_j \text{ for } 1 < i < s \text{ and } u < p \leq s. \quad (7)$$

PROOF. We have:

$$\frac{\partial^2 f_{i+1}}{\partial x_i \partial x_p} - \frac{\partial^2 f_i}{\partial x_i \partial x_p} = 0 - \frac{\partial^2 f_i}{\partial x_i \partial x_p} = -24x_u x_p (x_i^2 + \dots + x_s^2).$$

Thus the leading term is $x_i^3 x_p$. $\square$

LEMMA 3.7. *We consider the following $N \times N$ integer matrix:*

$$A_N = \begin{bmatrix} 5 & 1 & \dots & 1 & 1 \\ 1 & 5 & \dots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 5 & 1 \\ 1 & 1 & \dots & 1 & 5 \end{bmatrix}.$$

Then $\det(A_N) = (N+4)2^{N-2}$.

PROOF. By summing up the rows of the matrix A_N we obtain the following vector:

$$\mathbf{v} = \begin{bmatrix} (N+4) & \dots & (N+4) \end{bmatrix}.$$

For all $1 \leq i < N$, we subtract from the i -th row of A_N the vector $\frac{1}{N+4} \mathbf{v}$. Hence:

$$\det(A_N) = \begin{vmatrix} 4 & 0 & \dots & 0 & 0 \\ 0 & 4 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 4 & 0 \\ N+4 & N+4 & \dots & N+4 & N+4 \end{vmatrix} = (N+4)4^{N-1}$$

$\square$

LEMMA 3.8. *If the characteristic of $\mathbb{K}$ is larger than $s+4$, then $\mathcal{H}_2 = \langle x_i^2 h_i \mid 0 \leq i \leq s \text{ and } i \leq j \leq s \rangle$.*

PROOF. Clearly $\mathcal{H}_2 = \langle \frac{\partial^2 f_i}{\partial x_u^2} \mid 0 \leq i \leq s \text{ and } i \leq u \leq s \rangle$.

From the expression (6) of $\frac{\partial^2 f_i}{\partial x_u^2}$ we deduce that:

$$\begin{bmatrix} \frac{\partial^2 f_i}{\partial x_i^2} \\ \vdots \\ \frac{\partial^2 f_i}{\partial x_s^2} \end{bmatrix} = 6A_{s-i+1} \begin{bmatrix} x_i^2 h_i \\ \vdots \\ x_s^2 h_i \end{bmatrix}$$

Since the characteristic of $\mathbb{K}$ is $> s+4$, we know from lemma 3.7 that $\det(A_{s-i+1}) \neq 0$ and thus

$$\left\langle \frac{\partial^2 f_i}{\partial x_i^2}, \dots, \frac{\partial^2 f_i}{\partial x_s^2} \right\rangle = \langle x_i^2 h_i, \dots, x_s^2 h_i \rangle$$

$\square$

LEMMA 3.9. *If the characteristic of $\mathbb{K}$ is $> s+4$, it holds that $\mathcal{H}_2 = \langle x_0^4, x_0^2 x_1^2, x_1^4, x_0^2 x_2^2, x_1^2 x_2^2, x_2^4, \dots, x_0^2 x_s^2, x_1^2 x_s^2, \dots, x_{s-1}^2 x_s^2, x_s^4 \rangle$.*

PROOF. We set $\mathcal{J}_i = \langle x_i^2 h_i, \dots, x_s^2 h_i \rangle$. From lemma 3.8 we know that $\mathcal{H}_2 = \mathcal{J}_0 \cap \mathcal{J}_s$. We prove by induction that $\mathcal{J}_i \bmod \mathcal{J}_{i+1} \cap \dots \cap \mathcal{J}_s = \langle x_i^2 x_i^2, \dots, x_i^2 x_s^2 \rangle$.

For $i' = s$ the property is true since $\mathcal{J}_s = \langle x_s^4 \rangle$.

Now we assume that the property is true for all $i' > i$. This implies that for all $j > i$:

$$x_j^2 h_i = x_j^2 x_i^2 + \sum_{k=i+1}^s x_j^2 x_k^2 \longrightarrow_{\mathcal{J}_{i+1} \cap \dots \cap \mathcal{J}_s} x_j^2 x_i^2,$$

where $\longrightarrow_{\mathcal{J}}$ stands for the reduction modulo $\mathcal{J}$.

Finally $x_i^2 h_i = x_i^4 + \sum_{j=i+1}^s x_i^2 x_j^2 \longrightarrow_{\langle x_{i+1}^2 h_i, \dots, x_s^2 h_i \rangle} x_i^4$. Consequently the property is also true if $i' = i$. $\square$

We now summarize our results.

COROLLARY 3.2. *Let $f = g \circ h \in D_{r,3,2}$ be the particular example defined previously. If the characteristic of $\mathbb{K}$ is larger than $s+4$, the truncated 2-Gröbner basis of $\partial \mathcal{J}_f^2 : x_s^2$ is*

$$[x_0^2, \dots, x_s^2] = \mathcal{L}(h)$$

PROOF. According to the previous lemmas 3.5, 3.6, and 3.9, the leading terms of $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$ are pairwise distinct. We deduce a 4-Gröbner basis of $\partial \mathcal{J}_f^2$. Hence, the polynomials in $\partial \mathcal{J}_f^2$ of degree 4 divisible by x_s^2 are in $\mathcal{H}_3$. The result comes from the fact that these $s+1$ polynomials are the monomials $[x_0^2 x_s^2, \dots, x_s^2 x_s^2]$. $\square$

4. GENERIC UNIQUENESS OF THE LEFT COMPONENT

The left component of a decomposition can be recovered by solving a linear system as soon as h (or any basis of $\mathcal{L}(h)$) is known. Indeed, given f and h , a solution g to $f = g \circ h$ can be described by a system of linear equations. This system has

$$\alpha = (t+1) \binom{r+n}{r}$$

equations, each corresponding to one monomial in f . The coefficients in this linear system are polynomials in the coefficients of h . The unknowns correspond to the coefficients of g are

$$\beta = (t+1) \binom{s+\ell}{s}$$

in number. When can we expect g to be uniquely determined by f and h ? Generically, this corresponds to the question of whether $\alpha \geq \beta$.

THEOREM 4.1. 1. *If $s \leq r + \ell(m-1)$ and $\ell \leq r$, then $\alpha \geq \beta$.*

2. *If $s = r + \ell(m-1)$, $m \geq 2$, and $\ell \leq r$, then $\alpha \geq \beta$.*

3. *If $s > r + \ell(m-1)$ and $r \leq \ell$, then $\alpha < \beta$.*

4. *If $s \geq (r+n)(n+1)/(\ell+1) - \ell$, $m \geq 2$, and $\ell \leq r \leq 2\ell$, then $\alpha < \beta$.*

PROOF. (1) We have

$$\alpha \geq \beta \Leftrightarrow \frac{(r+n)^{\ell}}{r!} = \binom{r+n}{r} \geq \binom{s+\ell}{s} = \frac{(s+\ell)^{\ell}}{\ell!} \quad (8)$$

$$\Leftrightarrow (r+n)^{\ell} (r+n-\ell)^{r-\ell} \geq \frac{r!}{\ell!} (s+\ell)^{\ell} = (s+\ell)^{\ell} r^{r-\ell}, \quad (9)$$

where $x^{\ell} = x \cdot (x-1) \cdots (x-\ell+1)$ is the falling factorial (or Pochhammer symbol). We have $r+n-\ell = r + \ell(m-1) \geq r$ and $r+n \geq s+\ell$, so that the inequality (9) holds.

(2) Let $k = r + n = s + \ell$. We have $n \geq m\ell \geq 2\ell$, and

$$\begin{aligned} \alpha \geq \beta &\Leftrightarrow \binom{k}{r} \geq \binom{k}{s} \\ &\Leftrightarrow \frac{|r-n|}{2} = |r - \frac{k}{2}| \leq |s - \frac{k}{2}| \\ &= \frac{|2r+2n-2\ell-(r+n)|}{2} = \frac{|r+n-2\ell|}{2} = \frac{r+n-2\ell}{2} \\ &\Leftrightarrow |r-n| \leq n+r-2\ell. \end{aligned}$$

If $r \geq n$, then this holds since $0 \leq 2n-2\ell = 2\ell(m-1)$, and otherwise we have $|r-n| = n-r \leq n+r-2\ell$, since $\ell \leq r$.

(3) Similarly to (1), we write

$$\alpha < \beta \Leftrightarrow (r+n)^\ell (r+n-\ell)^{n-\ell} < (s+\ell)^\ell n^{n-\ell}.$$

Since $r \leq \ell$, the latter inequality is satisfied by assumption.

(4) We write

$$\frac{r!}{t+1} \alpha = (r+n)^r = (r+n) \cdots (n+1), \quad (10)$$

$$\frac{r!}{t+1} \beta = (s+\ell)^\ell \frac{r!}{\ell!} = (s+\ell)^\ell r^{r-\ell} \quad (11)$$

$$= (s+\ell) \cdots (s+1) \cdot r \cdots (\ell+1). \quad (12)$$

In both products, we multiply the first and last terms, the second and second last terms, etc. The resulting biproducts are $(r+n-i)(n+1+i)$ and $(s+\ell-i)(\ell+1+i)$, respectively, for $0 \leq i < r-\ell$. The assumption on s implies $s+\ell > r+n$, as in 3, since $(n+1)/(\ell+1) > 1$. In particular, we have $r < s$, and for $i \geq 0$

- $(r+n)(n+1) - \ell(\ell+1) \leq s(\ell+1)$,
- $(r+n)(n+1) - (s+\ell)(\ell+1) - i(s-r) < (r+n)(n+1) - (s+\ell)(\ell+1) \leq 0$,
- $(r+n-i)(n+1+i) \leq (s+\ell-i)(\ell+1+i)$.

Since $r-\ell \leq \ell$, the factors not absorbed in these $r-\ell$ biproducts are

- $(r+n-(r-\ell)) \cdots (n+1+r-\ell) = (n+\ell) \cdots (n+r-\ell+1)$ in (10),
- $(s+\ell-(r-\ell)) \cdots (s+1) = (s+2\ell-r) \cdots (s+1)$ in (12).

(These products are empty if $r = 2\ell$.) The assumption guarantees that $n+\ell-i < s+2\ell-r-i$ for $i \geq 0$, and $\alpha < \beta$ follows. $\square$

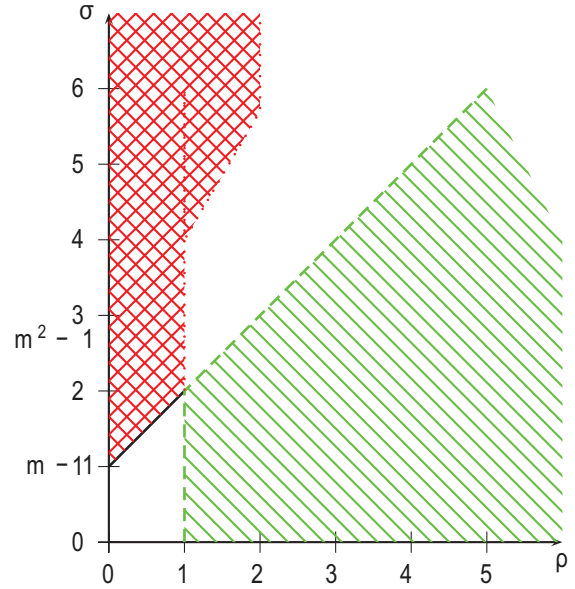
5. CONCLUSION

In order to visualize the result, we divide the variables by ℓ , obtaining $\rho = r/\ell$ and $\sigma = s/\ell$. In the figure on the opposite page, we have $\alpha \geq \beta$ in the green striped area, $\alpha < \beta$ in the red hashed area, and $\alpha = \beta$ on the diagonal line.

For our application, we think of ℓ and m (and hence n) as being fairly small, and of r and s as being substantially larger. Thus the right-hand striped area in the figure is relevant for us.

If $\alpha < \beta$, then the system for solving $f = g \circ h$ is underdetermined and has either no or many solutions. If $\alpha \geq \beta$, we have at least as many equations as unknowns. We conjecture that for a “generic” h , the system has maximal rank and thus is overdetermined. By trying to solve it, we determine whether a solution exists or not.

The central result of this paper is the proof in the preceding sections of this conjecture in the cases (2,2) and (3,2).



6. REFERENCES

- [1] V. S. Alagar and M. Thanh. *Fast Polynomial Decomposition Algorithms*. In Proc. EUROCAL85, Lecture Notes in Computer Science, vol. 204, pp. 150-153, Springer-Verlag, 1985.
- [2] M. Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie*. Thèse de doctorat, Université de Paris VI, 2004.
- [3] M. Bardet, J.-C. Faugère, and B. Salvy. *On the Complexity of Gröbner Basis Computation of Semi-Regular Overdetermined Algebraic Equations*. In Proc. of International Conference on Polynomial System Solving (ICPSS), pp. 71-75, 2004.
- [4] M. Bardet, J.-C. Faugère, B. Salvy and B.-Y. Yang. *Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems*. In Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry, 2005.
- [5] D. R. Barton and R. E. Zippel. *Polynomial decomposition algorithms*. J. Symb. Comp., 1, pp. 159-168, 1985.
- [6] B. Buchberger. *An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal (German)*, PhD Thesis, University of Innsbruck, Math. Institute, Austria, 1965. (English Translation: J.S.C., Special Issue on Logic, Mathematics, and Computer Science: Interactions. Vol. 41 (3-4), pp 475-511, 2006).
- [7] B. Buchberger. *Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems (An Algorithmical Criterion for the Solvability of Algebraic Systems of Equations)* Aequationes mathematicae 4/3, 1970, pp. 374-383. (English translation in: B. Buchberger, F. Winkler (eds.), Gröbner Bases and Applications, Proc. of the International Conference “33 Years of Gröbner Bases”, 1998, RISC, Austria, London Mathematical Society Lecture Note Series, Vol. 251, Cambridge University Press, 1998, pp. 535-545.)
- [8] B. Buchberger. *Gröbner Bases : an Algorithmic Method in Polynomial Ideal Theory*. Recent trends in multidimensional systems theory. Reider ed. Bose, 1985.

- [9] B. Buchberger, G.-E. Collins, and R. Loos. *Computer Algebra Symbolic and Algebraic Computation*. Springer-Verlag, second edition, 1982.
- [10] E.-W. Chionh, X.-S. Gao, L.-Y. Shen. *Inherently Improper Surface Parametric Supports*. Computer Aided Geometric Design 23 (2006), pp. 629–639.
- [11] D. A. Cox, J.B. Little, and D. O’Shea. *Ideals, Varieties, and Algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer-Verlag. New York, 1992.
- [12] M. Dickerson. *The functional Decomposition of Polynomials*. Ph.D Thesis, TR 89-1023, Departement of Computer Science, Cornell University, Ithaca, NY, July 1989.
- [13] M. Dickerson. *General Polynomial Decomposition and the s-1-decomposition are NP-hard*. International Journal of Foundations of Computer Science, 4:2 (1993), pp. 147–156.
- [14] F. Dorey and G. Whaples. *Prime and composite polynomials*. J. Algebra, (28), pp. 88-101, 1974.
- [15] J.-C. Faugère. *A New Efficient Algorithm for Computing Gröbner Basis without Reduction to Zero: F5*. Proceedings of ISSAC, pp. 75–83. ACM press, July 2002.
- [16] J.-C. Faugère, L. Perret. *Cryptanalysis of $2R^-$ schemes*. Advances in Cryptology – CRYPTO 2006, Lecture Notes in Computer Science, vol. 4117, pp. 357–372, Springer-Verlag, 2006.
- [17] J.-C. Faugère, L. Perret. *An Efficient Algorithm for Decomposing Multivariate Polynomials and its Applications to Cryptography*. Special Issue of JSC, “Gröbner Bases techniques in Coding Theory and Cryptography”, on-line available.
- [18] J.-C. Faugère, L. Perret. *High order derivatives and decomposition of multivariate polynomials*. Proceedings of ISSAC, pp. 207-214. ACM press, July 2009.
- [19] R. Fröberg. *An inequality for Hilbert series of graded algebras*. Math. Scand., 56(2) :117–144, 1985.
- [20] J. von zur Gathen. *The number of decomposable univariate polynomials*. Proceedings of ISSAC, pp. 359-366. ACM press, July 2009.
- [21] J. von zur Gathen. *Functional decomposition of polynomials: the tame case*. J. Symb. Comput. (9), pp. 281–299, 1990.
- [22] J. von zur Gathen. *Functional decomposition of polynomials: the wild case*. J. Symb. Comput. (10), pp. 437–452, 1990.
- [23] J. von zur Gathen, J. Gutierrez, R. Rubio. *Multivariate Polynomial Decomposition*. Applicable Algebra in Engineering, Communication and Computing, 14 (1), pp. 11–31, 2003.
- [24] J. Gutierrez, D. Sevilla. *Computation of Unirational fields*. J. Symb. Comput. 41(11), pp. 1222–1244, 2006.
- [25] J. Gutierrez, R. Rubio, D. Sevilla. *On Multivariate Rational Function Decomposition*. J. Symb. Comput. 33(5), pp. 545–562, 2002.
- [26] D. Kozen, and S. Landau. *Polynomial Decomposition Algorithms*. J. Symb. Comput. (7), pp. 445–456, 1989.
- [27] J. F. Ritt. *Prime and Composite Polynomials*. Trans. Amer. Math. Soc., (23), pp 51-66, 1922.
- [28] M. Sweedler. *Using Gröbner Bases to Determine the Algebraic and Transcendental Nature of Field Extensions: Return of the Killer Tag Variables*. Proc. AAECC, 66–75, 1993.
- [29] S. M. Watt. *Functional Decomposition of Symbolic Polynomials*. In Proc. International Conference on Computational Sciences and its Applications, (ICCSA 2008), IEEE Computer Society, pp. 353–362.
- [30] D.F. Ye, Z.D. Dai and K.Y. Lam. *Decomposing Attacks on Asymmetric Cryptography Based on Mapping Compositions*, Journal of Cryptology (14), pp. 137–150, 2001.

NumGfun: a Package for Numerical and Analytic Computation with D-finite Functions

Marc Mezzarobba

Algorithms Project-Team, INRIA Paris-Rocquencourt, France
marc.mezzarobba@inria.fr

ABSTRACT

This article describes the implementation in the software package NumGfun of classical algorithms that operate on solutions of linear differential equations or recurrence relations with polynomial coefficients, including what seems to be the first general implementation of the fast high-precision numerical evaluation algorithms of Chudnovsky & Chudnovsky. In some cases, our descriptions contain improvements over existing algorithms. We also provide references to relevant ideas not currently used in NumGfun.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms

General Terms

Algorithms, Experimentation, Theory

Keywords

D-finite functions, linear differential equations, certified numerical computation, bounds, Maple

1. INTRODUCTION

Support for computing with *D-finite functions*, that is, solutions of linear differential equations with polynomial coefficients, has become a common feature of computer algebra systems. For instance, Mathematica now provides a data structure called `DifferentialRoot` to represent arbitrary D-finite functions by differential equations they satisfy and initial values. Maple's `DEsol` is similar but more limited.

An important source of such general D-finite functions is combinatorics, due to the fact that many combinatorial structures have D-finite generating functions. Moreover, powerful methods allow to get from a combinatorial description of a class of objects to a system of differential equations that “count” these objects, and then to extract precise asymptotic information from these equations, even when no explicit

counting formula is available [15, 30]. A second major application of D-finiteness is concerned with special functions. Indeed, many classical functions of mathematical physics are D-finite (often by virtue of being defined as “interesting” solutions of simple differential equations), which allows to treat them uniformly in algorithms. This is exploited by the *Encyclopedia of Special Functions* [25] and its successor under development, the *Dynamic Dictionary of Mathematical Functions* [20], an interactive computer-generated handbook of special functions.

These applications require at some point the ability to perform “analytic” computations with D-finite functions, starting with their numerical evaluation. Relevant algorithms exist in the literature. In particular, D-finite functions may be computed with an absolute error bounded by 2^{-n} in $n \log^{O(1)} n$ bit operations—that is, in softly linear time in the size of the result written in fixed-point notation—at any point of their Riemann surfaces [12], the necessary error bounds also being computed from the differential equation and initial values [32]. However, these algorithms have remained theoretical [13, §9.2.1]. The ability of computer algebra systems to work with D-finite functions is (mostly) limited to symbolic manipulations, and the above-mentioned fast evaluation algorithm has served as a recipe to write numerical evaluation routines for specific functions rather than as an algorithm for the entire class of D-finite functions.

This article introduces NumGfun, a Maple package that attempts to fill this gap, and contains, among other things, a general implementation of that algorithm. NumGfun is distributed as a subpackage of `gfun` [29], under the GNU LGPL. Note that it comes with help pages: the goal of the present article is not to take the place of user documentation, but rather to describe the features and implementation of the package, with supporting examples, while providing an overview of techniques relevant to the development of similar software. The following examples illustrate typical uses of NumGfun, first to compute a remote term from a combinatorial sequence, then to evaluate a special function to high precision near one of its singularities.

Example 1. The Motzkin number M_n is the number of ways of drawing non-intersecting chords between n points placed on a circle. Motzkin numbers satisfy $(n+4)M_{n+2} = 3(n+1)M_n + (2n+5)M_{n+1}$. Using NumGfun, the command `nth_term({(n+4)*M(n+2)=3*(n+1)*M(n)+(2*n+5)*M(n+1), M(0)=1,M(1)=1},M(n),k)` computes $M_{10^5} = 6187 \dots 7713 \simeq 10^{47706}$ in 4.7 s and $M_{10^6} = 2635 \dots 9151 \simeq 10^{477112}$ in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

¹All timings reported in this article were obtained with the following configuration: Intel T7250 CPU at 2 GHz, 1 GB of

1 min. Naïvely unrolling the recurrence (using Maple) takes 10.7 s for M_{10^5} , and 41 s for $M_{2 \cdot 10^5}$. On this (non-generic) example, `nth_term` could be made competitive for smaller indices by taking advantage of the fact that the divisions that occur while unrolling the recurrence are exact.

Example 2. The double confluent Heun function $U_{\alpha,\beta,\gamma,\delta}$ satisfies $(z^2 - 1)^3 U''(z) + (2z^5 - 4z^3 - \alpha z^4 + 2z + \alpha) U'(z) + (\beta z^2 + (2\alpha + \gamma)z + \delta) U(z) = 0$, $U(0) = 1$, $U'(0) = 0$. It is singular at $z = \pm 1$. The command `evaldiffeq(eq,y(z), -0.99,1000)` where `eq` is this differential equation yields $U_{1,\frac{1}{3},\frac{1}{2},3}(-0.99) \approx 4.67755\dots(990 \text{ digits})\dots 05725$ in 22 s.

Related work. Most algorithms implemented in NumGfun originate in work of Chudnovsky & Chudnovsky and of van der Hoeven. Perhaps the most central of these is the “bit burst” numerical evaluation method [12]. It belongs to the family of binary splitting algorithms for D-finite series, hints at the existence of which go back to [2, §178], and generalizes earlier work of Brent [6] for specific elementary functions. Better known (thanks among others to [21]) binary splitting algorithms can be seen as special cases of the bit burst algorithm. One reason why, unlike these special cases, it was not used in practice is that in [12], none of the error control needed to ensure the accuracy of the computed result is part of the algorithm. Van der Hoeven’s version [32] addresses this issue, thus giving a full-fledged evaluation algorithm for the class of D-finite functions, as opposed to a method to compute any D-finite function given certain bounds.

These algorithms extend to the computation of limits of D-finite functions at singularities of their defining equation. The case of regular singularities is treated both in [11, 12], and more completely in [33], that of irregular singular points in [35]. See [4, §12.7], [35, §1] for more history and context.

On the implementation side, routines based on binary splitting for the evaluation of various elementary and special functions are used in general-purpose libraries such as CLN [19] and MPFR [16, 23]. Binary splitting of fast converging series is also the preferred algorithm of software dedicated to the high-precision computation of mathematical constants on standard PCs, including the current record holder for π [3]. Finally, even the impressive range of built-in functions of computer algebra systems is not always sufficient for applications. Works on the implementation of classes of “less common” special functions that overlap those considered in NumGfun include [1, 14].

This work is based in part on the earlier [26].

Contribution. The main contribution presented in this article is NumGfun itself. We recall the algorithms it uses, and discuss various implementation issues. Some of these descriptions include improvements or details that do not seem to have appeared elsewhere. Specifically: (i) we give a new variant of the analytic continuation algorithm for D-finite functions that is faster with respect to the order and degree of the equation; (ii) we improve the complexity analysis of the bit burst algorithm by a factor $\log \log n$; (iii) we point out that Poole’s method to construct solutions of differential equations at regular singular points can be rephrased in a compact way in the language of noncommutative polynomials, leading to faster evaluation of D-finite functions in these points; and (iv) we describe in some detail the practical

computation of all the bounds needed to obtain a provably correct result.

What NumGfun is not. Despite sharing some of the algorithms used to compute mathematical constants to billions of digits, our code aims to cover as much as possible of the class of D-finite functions, not to break records. Also, it is limited to “convergent” methods: asymptotic expansions, summation to the least term, and resummation of divergent power series are currently out of scope.

Terminology. Like the rest of gfun, NumGfun works with *D-finite functions* and *P-recursive sequences*. We recall only basic definitions here; see [30, §6.4] for further properties. A formal power series $y \in \mathbb{C}[[z]]$ is *D-finite* over $K \subseteq \mathbb{C}$ if it solves a non-trivial linear differential equation

$$y^{(r)}(z) + a_{r-1}(z) y^{(r-1)}(z) + \dots + a_0(z) y(z) = 0 \quad (1)$$

with coefficients $a_k \in K(z)$. The same definition applies to analytic functions. A sequence $u \in \mathbb{C}^{\mathbb{N}}$ is *P-recursive* over K if it satisfies a non-trivial linear recurrence relation

$$u_{n+s} + b_{s-1}(n) u_{n+s-1} + \dots + b_0(n) u_n = 0, \quad b_k \in K(n). \quad (2)$$

A sequence $(u_n)_{n \in \mathbb{N}}$ is P-recursive if and only if its generating series $\sum_{n \in \mathbb{N}} u_n z^n$ is D-finite.

The poles of the coefficients a_k of (1) are its *singular points*; nonsingular points are called *ordinary*. In gfun, a D-finite function is represented by a differential equation of the form (1) and initial values at the origin, which we assume to be an ordinary point. Similarly, P-recursive sequences are encoded by a recurrence relation plus initial values, as in Ex. 1 above. If $y(z) = \sum_{k=0}^{\infty} y_k z^k$, we let $y_n(z) = \sum_{k=0}^{n-1} y_k z^k$ and $y_n(z) = \sum_{k=n}^{\infty} y_k z^k$.

The *height* of an object is the maximum bit-size of the integers appearing in its representation: the height of a rational number p/q is $\max(\lceil \log p \rceil, \lceil \log q \rceil)$, and that of a complex number (we assume that elements of number fields $\mathbb{Q}(\zeta)$ are represented as $\sum_i x_i \zeta^i / d$ with $x_i, d \in \mathbb{Z}$), polynomial, matrix, or combination thereof with rational coefficients is the maximum height of its coefficients. We assume that the bit complexity $M(n)$ of n -bit integer multiplication satisfies $M(n) = n(\log n)^{O(1)}$, $M(n) = \Omega(n \log n)$, and $M(n+m) \geq M(n) + M(m)$, and that $s \times s$ matrices can be multiplied in $O(s^\omega)$ operations in their coefficient ring.

2. BINARY SPLITTING

“Unrolling” a recurrence relation of the form (2) to compute $u_0, \dots, u_N$ takes $\Theta(N^2 M(\log N))$ bit operations, which is almost linear in the total size of $u_0, \dots, u_N$, but quadratic in that of u_N . The binary splitting algorithm computes a single term u_N in essentially linear time, as follows: (2) is first reduced to a matrix recurrence of the first order with a single common denominator:

$$q(n) U_{n+1} = B(n) U_n, \quad B(n) \in \mathbb{Z}[n]^{s \times s}, q(n) \in \mathbb{Z}[n], \quad (3)$$

so that $U_N = P(0, N) U_0 / (\prod_{i=0}^{N-1} q(i))$, where $P(j, i) = B(j-1) \dots B(i+1) B(i)$. One then computes $P(0, N)$ recursively as $P(0, N) = P(\lfloor N/2 \rfloor, N) P(0, \lfloor N/2 \rfloor)$, and the denominator $\prod_{i=0}^{N-1} q(i)$ in a similar fashion (but separately, in order to avoid expensive gcd computations).

The idea of using product trees to make the most of fast multiplication dates back at least to the seventies [4, §12.7].

The general statement below is from [12, Theorem 2.2], except that the authors seem to have overlooked the cost of evaluating the polynomials at the leaves of the tree.

THEOREM 1 (CHUDNOVSKY, CHUDNOVSKY). *Let u be a P -recursive sequence over $\mathbb{Q}(i)$, defined by (2). Assume that the coefficients $b_k(n)$ of (2) have no poles in $\mathbb{N}$. Let d, h denote bounds on their degrees (of numerators and denominators) and heights, and d', h' corresponding bounds for the coefficients of $B(n)$ and $q(n)$ in (3). Assuming $N \gg s, d$, the binary splitting algorithm outlined above computes one term u_N of u in $O(s^\omega M(N(h' + d' \log N)) \log(Nh'))$, that is, $O(s^\omega M(sdN(h + \log N)) \log(Nh))$, bit operations.*

PROOF SKETCH. Write $H = h' + d' \log N$. Computing the product tree $P(0, N)$ takes $O(s^\omega M(NH) \log N)$ bit operations [12, §2] (see also Prop. 1 below), and the evaluation of each leaf $B(i)$ may be done in $O(M(H) \log d')$ operations [5, §3.3]. This gives u_N as a fraction that is simplified in $O(M(NH) \log(NH))$ operations [8, §1.6].

Now consider how (2) is rewritten into (3). With coefficients in $\mathbb{Z}[i]$ rather than $\mathbb{Q}(i)$, the $b_k(n)$ have height $h'' \leq (d+1)h$. To get $B(n)$ and $q(n)$, it remains to reduce to common denominator the whole equation; hence $d' \leq sd$ and $h' \leq s(h'' + \log s + \log d)$. These two conversion steps take $O(M(sdh \log^2 d))$ and $O(M(d'h' \log s))$ operations respectively, using product trees. The assumption $d, s = o(N)$ allows to write $H = O(sd(h + \log N))$ and get rid of some terms, so that the total complexity simplifies as stated. $\square$

Since the height of u_N may be as large as $\Omega((N+h) \log N)$, this result is optimal with respect to h and N , up to logarithmic factors. The same algorithm works over any algebraic number field instead of $\mathbb{Q}(i)$. This is useful for evaluating D-finite functions “at singularities” (§4). More generally, similar complexity results hold for product tree computations in torsion-free $\mathbb{Z}$ -algebras (or $\mathbb{Q}$ -algebras: we then write $A = \mathbb{Q} \otimes_{\mathbb{Z}} A'$ for some $\mathbb{Z}$ -algebra A' and multiply in $\mathbb{Z} \times A'$), keeping in mind that, without basis choice, the height of an element is defined only up to some additive constant.

Constant-factor improvements. Several techniques permit to improve the constant hidden in the $O(\cdot)$ of Theorem 1, by making the computation at each node of the product tree less expensive. We consider two models of computation.

In the *FFT model*, we assume that the complexity of long multiplication decomposes as $M(n) = 3F(2n) + O(n)$, where $F(n)$ is the cost of a discrete Fourier transform of size n (or of another related linear transform, depending on the algorithm). FFT-based integer multiplication algorithms adapt to reduce the multiplication of two matrices of height n in $\mathbb{Z}^{s \times s}$ to $O(n)$ multiplications of matrices of height $O(1)$, for a total of $O(s^2 M(n) + s^\omega n)$ bit operations. This is known as “FFT addition” [4], “computing in the FFT mode” [32], or “FFT invariance”. A second improvement (“FFT doubling”, attributed to R. Kramer in [4, §12.8]) is specific to the computation of product trees. The observation is that, at an internal node where operands of size n get multiplied using three FFTs of size $2n$, every second coefficient of the two direct DFTs is already known from the level below.

The second model is *black-box multiplication*. There, we may use fast multiplication formulae that trade large integer multiplications for additions and multiplications by constants. The most obvious example is that the products in $\mathbb{Q}(i)$ may be done in four integer multiplications using Karatsuba’s formula instead of five with the naïve algorithm. In general,

elements of height h of an algebraic number field of degree d may be multiplied in $2dM(h) + O(h)$ bit operations using the Toom-Cook algorithm. The same idea applies to the matrix multiplications. Most classical matrix multiplication formulae such as Strassen’s are so-called bilinear algorithms. Since we are working over a commutative ring, we may use more general *quadratic* algorithms [9, §14.1]. In particular, for all s , Waksman’s algorithm [38] multiplies $s \times s$ matrices over a commutative ring R using $s^2 \lceil s/2 \rceil + (2s-1) \lfloor s/2 \rfloor$ multiplications in R , and Makarov’s [24] multiplies 3×3 matrices in 22 scalar multiplications. These formulas alone or as the base case of a Strassen scheme achieve what seems to be the best known multiplication count for matrices of size up to 20.

Exploiting these ideas leads to the following refinement of Theorem 1. Similar results can be stated for general $\mathbb{Z}$ -algebras, using their rank and multiplicative complexity [9].

PROPOSITION 1. *Let d' and h' denote bounds on the degrees and heights (respectively) of $B(n)$ and $q(n)$ in Eq. (3). As $N, h' \rightarrow \infty$ (s and d' being fixed), the number of bit operations needed to compute the product tree $P(0, N)$ is at most $(C + o(1))M(N(h' + d' \log N) \log(Nh'))$, with $C = (2s^2 + 1)/6$ in the FFT model, and $C = (3\text{MM}(s) + 1)/4$ in the black-box model. Here $\text{MM}(s) \leq (s^3 + 3s^2)/2$ is the algebraic complexity of $s \times s$ matrix multiplication over $\mathbb{Z}$.*

PROOF. Each node at the level k (level 0 being the root) of the tree essentially requires multiplying $s \times s$ matrices with entries in $\mathbb{Z}[i]$ of height $H_k = N(h' + d' \log N)/2^{k+1}$, plus denominators of the same height. In the FFT model, this may be done in $(2s^2 + 1)M(H_k)$ operations. Since we assume $M(n) = \Omega(n \log n)$, we have $\sum_{k=0}^{\lceil \log N \rceil} 2^k M(H_k) \leq \frac{1}{2} M(H_0)$ (a remark attributed to D. Stehlé in [39]). Kramer’s trick saves another factor $\frac{3}{2}$. In the black-box model, the corresponding cost for one node is $(3\text{MM}(s) + 1)M(H_k)$ with Karatsuba’s formula. Stehlé’s argument applies again. $\square$

Note that the previous techniques save time only for dense objects. In particular, one should not use the “fast” matrix multiplication formulae in the few bottom levels of product trees associated to recurrences of the form (3), since the matrices at the leaves are companion.

Continuing on this remark, these matrices often have some structure that is preserved by successive multiplications. For instance, let $s_n = \sum_{k=0}^{n-1} u_k$ where (u_n) satisfies (2). It is easy to compute a recurrence and initial conditions for (s_n) and go on as above. However, unrolling the recurrences (2) and $s_{n+1} - s_n = u_n$ simultaneously as

$$\begin{pmatrix} u_{n+1} \\ \vdots \\ u_{n+s-1} \\ u_{n+s} \\ s_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \\ * & * & \cdots & * & 0 \\ 1 & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} u_n \\ \vdots \\ u_{n+s-2} \\ u_{n+s-1} \\ s_n \end{pmatrix} \quad (4)$$

is more efficient. Indeed, all matrices in the product tree for the numerator of (4) then have a rightmost column of zeros, except for the value in the lower right corner, which is precisely the denominator. With the notation $\text{MM}(s)$ of Proposition 1, each product of these special matrices uses $\text{MM}(s) + s^2 + s + 1$ multiplications, *vs.* $\text{MM}(s+1) + 1$ for the dense variant. Hence the formula (4) is more efficient as soon as $\text{MM}(s+1) - \text{MM}(s) \geq s(s+1)$, which is true both for the naïve multiplication algorithm and for Waksman’s algorithm (compare [39]). In practice, on Ex. 3 below, if

one puts $u_n = s_{n+1} - s_n$ in (2, 3) instead of using (4), the computation time grows from 1.7 s to 2.7 s.

The same idea applies to any recurrence operator that can be factored. Further examples of structure in product trees include even and odd D-finite series (e.g., [8, §4.9.1]). In all these cases, the naïve matrix multiplication algorithm automatically benefits from the special shape of the problem (because multiplications by constants have negligible cost), while fast methods must take it into account explicitly.

Remark 1. A weakness of binary splitting is its comparatively large space complexity $\Omega(n \log n)$. Techniques to reduce it are known and used by efficient implementations in the case of recurrences of the first order [10, 17, 19, 3].

Implementation. The implementation of binary splitting in NumGfun includes some of the tricks discussed in this section. FFT-based techniques are currently not used because they are not suited to implementation in the Maple language. This implementation is exposed through two user-level functions, `nth_term` and `fnth_term`, that allow to evaluate P-recursive sequences (`fnth_term` replaces the final gcd by a less expensive division and returns a floating-point result). Additionally, `gfun[rectoproc]`, which takes as input a recurrence and outputs a procedure that evaluates its solution, automatically calls the binary splitting code when relevant. Examples 1 and 3 illustrate the use of these functions on integer sequences and convergent series respectively.

Example 3. [7, §6.10] Repeated integration by parts of the integral representation of Γ yields for $0 < \text{Re}(z) < 1$

$$\Gamma(z) = \sum_{n=0}^{\infty} \frac{e^{-t} t^{n+z}}{z(z+1) \cdots (z+n)} + \int_t^{\infty} e^{-u} u^{z-1} du.$$

Taking $t = 29^3$, it follows that the sum $\sum_{n=0}^{65000} u_n$, where $u_0 = 87e^{-t}$ and $(3n+4)u_{n+1} = 3tu_n$, is within 10^{-10000} of $\Gamma(1/3)$, whence $\Gamma(1/3) \simeq 2.67893 \dots (9990 \text{ digits}) \dots 99978$. This computation takes 1.7 s.

3. HIGH-PRECISION EVALUATION OF D-FINITE FUNCTIONS

We now recall the numerical evaluation algorithms used in NumGfun, and discuss their implementation.

Let $y(z) = \sum_n y_n z^n$ be a D-finite series with radius of convergence ρ at the origin. Let $z \in \mathbb{Q}(i)$ be such that $|z| < \rho$ and $\text{height}(z) \leq h$. The sequence $(y_n z^n)$ is P-recursive, so that the binary splitting algorithm yields a fast method for the high-precision evaluation of $y(z)$. Here “high-precision” means that we let the precision required for the result go to infinity in the complexity analysis of the algorithm.

More precisely, $(y_n z^n)$ is canceled by a recurrence relation of height $O(h)$. By Theorem 1, $y(z)$ may hence be computed to the precision 10^{-p} in

$$O(M(N(h + \log N)) \log(Nh)) \quad (5)$$

bit operations, where N is chosen so that $|y_N(z)| \leq 10^{-p}$, i.e. $N \sim p/\log(\rho/|z|)$ if $\rho < \infty$, and $N \sim p/(\tau \log p)$ for some τ if $\rho = \infty$.

In practice, the numbers of digits of (absolute) precision targeted in NumGfun range from the hundreds to the millions. Accuracies of this order of magnitude are useful in some applications to number theory [12], and in heuristic equality tests [33, §5]. It can also happen that the easiest way to obtain a moderate-precision output involves high-precision

intermediate computations, especially when the correctness of the result relies on pessimistic bounds.

Analytic continuation. Solutions of the differential equation (1) defined in the neighborhood of 0 extend by analytic continuation to the universal covering of $\mathbb{C} \setminus S$, where S is the (finite) set of singularities of the equation. D-finite functions may be evaluated fast at any point by a numerical version of the analytic continuation process that builds on the previous algorithm [12]. Rewrite (1) in matrix form

$$Y'(z) = A(z)Y(z), \quad Y(z) = \left(\frac{y^{(i)}(z)}{i!} \right)_{0 \leq i < r}. \quad (6)$$

This choice of $Y(z)$ induces, for all $z_0 \in \mathbb{C} \setminus S$, that of a family of *canonical solutions* of (1) defined by

$$y[z_0, j](z) = (z - z_0)^j + O((z - z_0)^r), \quad 0 \leq j < r,$$

that form a basis of the solutions of (1) in the neighborhood of z_0 . Stated otherwise, the vector $\mathbf{y}[z_0] = (y[z_0, j])_{0 \leq j < r}$ of canonical solutions at z_0 is the first row of the fundamental matrix $\mathbf{Y}[z_0](z)$ of (6) such that $\mathbf{Y}[z_0](z_0) = \text{Id}$.

By linearity, for any path $z_0 \rightarrow z_1$ in $\mathbb{C} \setminus S$, there exists a *transition matrix* $M_{z_0 \rightarrow z_1}$ that “transports initial conditions” (and canonical solutions) along the path:

$$Y(z_1) = M_{z_0 \rightarrow z_1} Y(z_0), \quad \mathbf{y}[z_0](z) = \mathbf{y}[z_1](z) M_{z_0 \rightarrow z_1}. \quad (7)$$

This matrix is easy to write out explicitly:

$$M_{z_0 \rightarrow z_1} = \mathbf{Y}[z_0](z_1) = \left(\frac{1}{i!} y[z_0, j]^{(i)}(z_1) \right)_{0 \leq i, j < r}, \quad (8)$$

evaluations at z_1 being understood to refer to the analytic continuation path $z_0 \rightarrow z_1$. Transition matrices compose:

$$M_{z_0 \rightarrow z_1 \rightarrow z_2} = M_{z_1 \rightarrow z_2} M_{z_0 \rightarrow z_1}, \quad M_{z_1 \rightarrow z_0} = M_{z_0 \rightarrow z_1}^{-1}. \quad (9)$$

NumGfun provides functions to compute M_γ for a given path γ (`transition_matrix`), and to evaluate the analytic continuation of a D-finite function along a path starting at 0 (`evaldifffeq`). In both cases, the path provided by the user is first subdivided into a new path $z_0 \rightarrow z_1 \rightarrow \cdots \rightarrow z_m$, $z_\ell \in \mathbb{Q}(i)$, such that, for all ℓ , the point $z_{\ell+1}$ lies within the disk of convergence of the Taylor expansions at z_ℓ of all solutions of (1). Using (8), approximations $\tilde{M}_\ell \in \mathbb{Q}(i)^{r \times r}$ of $M_{z_\ell \rightarrow z_{\ell+1}}$ are computed by binary splitting.

More precisely, we compute all entries of $\tilde{M}_\ell$ at once, as follows. For a generic solution y of (1), the coefficients u_n of $u(z) = y(z_\ell + z) = \sum_{n=0}^{\infty} u_n z^n$ are canceled by a recurrence of order s . Hence the partial sums $u_{;n}(z)$ of u satisfy

$$\begin{pmatrix} U_{n+1} \\ u_{;n+1}(z) \end{pmatrix} = \underbrace{\begin{pmatrix} C(n)z & 0 \\ K & 1 \end{pmatrix}}_{B(n)} \begin{pmatrix} U_n \\ u_{;n}(z) \end{pmatrix}, \quad (10)$$

where $K = (1, 0, \dots, 0)$ and $C(n) \in \mathbb{Q}(n)^{s \times s}$. Introducing an indeterminate δ , we let $z = z_{\ell+1} - z_\ell + \delta \in \mathbb{Q}(i)[\delta]$ and compute $B(N-1) \cdots B(0) \bmod \delta^r$ by binary splitting (an idea already used in [32]), for some suitable N . The upper left blocks of the subproducts are *kept factored* as a power of z times a matrix independent on z . In other words, clearing denominators, the computation of each subproduct

$$P = \frac{1}{d} \begin{pmatrix} D \cdot p & 0 \\ R & d \end{pmatrix} = P_{\text{high}} P_{\text{low}} \quad (p = \text{numer}(z)^m)$$

is performed as $D \leftarrow D_{\text{high}} D_{\text{low}}$; $p \leftarrow p_{\text{high}} p_{\text{low}}$; $R \leftarrow p_{\text{low}}(R_{\text{high}} C_{\text{low}}) + d_{\text{high}} R_{\text{low}}$; $d \leftarrow d_{\text{high}} d_{\text{low}}$. The powers of z can be computed faster, but the saving is negligible. Applying the row submatrix R of the full product to the matrix $U_0 = (\frac{1}{i!} y[z_\ell, j]^{(i)}(z_\ell))_{0 \leq i < s, 0 \leq j < r}$ yields a row vector equal to $(y[z_\ell, j]^{(i)}(z_{\ell+1} + \delta))_{0 \leq j < r} + O(\delta^r)$, each entry of which is a truncated power series whose coefficients are the entries of the corresponding column of $\tilde{M}_\ell$. This way of computing $\tilde{M}_\ell$ is roughly $\min(r, s^{\omega-1})$ times more efficient than the fastest of the variants from [12, 32].

In the function `transition_matrix`, we then form the product $\tilde{M}_{m-1} \cdots \tilde{M}_0$, again by binary splitting. In the case of `evaldiffeq`, we compute only the first row $\tilde{R}$ of $\tilde{M}_{m-1}$, and we form the product $\tilde{R} \tilde{M}_{m-2} \cdots \tilde{M}_0 \tilde{I}$, where $\tilde{I}$ is an approximation with coefficients in $\mathbb{Q}(i)$ of the vector $Y(z_0)$ of initial conditions (or this vector itself, if it has symbolic entries). The whole computation is done on unsimplified fractions, controlling all errors to guarantee the final result. Let us stress that no numerical instability occurs since all numerical operations are performed on rational numbers. We postpone the discussion of approximation errors (including the choice of N) to §5.

Example 4. Transition matrices corresponding to paths that “go round” exactly one singularity once are known as local monodromy matrices. A simple example is that of the equation $(1 + z^2)y'' + 2zy' = 0$, whose solution space is generated by 1 and $\arctan z$. Rather unsurprisingly, around i :

```
> transition_matrix((1+z^2)*diff(y(z),z),y(z),z)
+2*z*diff(y(z),z), y(z), [0,I+1,2*I,I-1,0], 20);
(1.00000000000000000000000000000000  3.14159265358979323846)
(0 1.00000000000000000000000000000000)
```

More generally, expressing them as entries of monodromy matrices is a way to compute many mathematical constants.

Another use of analytic continuation appears in Ex. 2: there, despite the evaluation point lying within the disk of convergence, NumGfun performs analytic continuation along the path $0 \rightarrow \frac{-1}{2} \rightarrow \frac{-3}{4} \rightarrow \frac{-22}{25} \rightarrow \frac{-47}{50} \rightarrow \frac{-99}{100}$ to approach the singularity more easily (cf. [12, Prop. 3.3]).

The “bit burst” algorithm. The complexity result (5) is quasi-optimal for $h = O(\log p)$, but becomes quadratic in p for $h = \Theta(p)$, which is the size of the approximation $\tilde{z} \in \mathbb{Q}(i)$ needed to compute $y(z)$ for arbitrary $z \in \mathbb{C}$. This issue can be avoided using analytic continuation to approach z along a path $z_0 \rightarrow z_1 \rightarrow \cdots \rightarrow z_m = \tilde{z}$ made of approximations of z whose precision grow geometrically:

$$|z_{\ell+1} - z_\ell| \leq 2^{-2^\ell}, \quad \text{height}(z_\ell) = O(2^\ell), \quad (11)$$

thus balancing h and $|z|$. This idea is due to Chudnovsky and Chudnovsky [12], who called it the bit burst algorithm, and independently to van der Hoeven with a tighter complexity analysis [32, 37]. The following proposition improves this analysis by a factor $\log \log p$ in the case where y has a finite radius of convergence. No similar improvement seems to apply to entire functions.

PROPOSITION 2. *Let y be a D -finite function. One may compute a 2^{-p} -approximation of y at a point $z \in \mathbb{Q}(i)$ of height $O(p)$ in $O(M(p \log^2 p))$ bit operations.*

PROOF. By (5) and (11), the step $z_\ell \rightarrow z_{\ell+1}$ of the bit-burst algorithm takes $O(M(p(2^\ell + \log p) \log p / 2^\ell))$ bit opera-

tions. Now

$$\sum_{\ell=0}^m M\left(\frac{p(2^\ell + \log p)}{2^\ell}\right) \log p \leq M\left(mn \log p + \sum_{\ell=0}^m \frac{p \log^2 p}{2^\ell}\right)$$

and $m = O(\log p)$, hence the total complexity. $\square$

Example 5. Consider the D -finite function y defined by the following equation, picked at random:

$$\begin{aligned} & \left(\frac{5}{12} - \frac{1}{4}z + \frac{19}{24}z^2 - \frac{5}{24}z^3\right)y^{(4)} + \left(-\frac{7}{24} + \frac{2}{3}z + \frac{13}{24}z^2 + \frac{1}{12}z^3\right)y''' \\ & + \left(\frac{7}{12} - \frac{19}{24}z + \frac{1}{8}z^2 + \frac{1}{3}z^3\right)y'' + \left(-\frac{3}{4} + \frac{5}{12}z + \frac{5}{6}z^2 + \frac{1}{2}z^3\right)y' \\ & + \left(\frac{5}{24} + \frac{23}{24}z + \frac{7}{8}z^2 + \frac{1}{3}z^3\right)y = 0, \\ & y(0) = \frac{1}{24}, y'(0) = \frac{1}{12}, y''(0) = \frac{5}{24}, y'''(0) = \frac{5}{24}. \end{aligned}$$

The singular points are $z \approx 3.62$ and $z \approx 0.09 \pm 0.73i$. By analytic continuation (along a path $0 \rightarrow \pi i$ homotopic to a segment) followed by bit-burst evaluation, we obtain

$$y(\pi i) \approx -0.52299\dots(990 \text{ digits})\dots53279 - 1.50272\dots90608i$$

after about 5 min. This example is among the “most general” that NumGfun can handle. Intermediate computations involve recurrences of order 8 and degree 17.

4. REGULAR SINGULAR POINTS

The algorithms of the previous section extend to the case where the analytic continuation path passes through *regular singular points* of the differential equation [11, 12, 33]. Work is in progress to support this in NumGfun, with two main applications in view, namely special functions (such as Bessel functions) defined using their local behaviour at a regular singular point, and “connection coefficients” arising in analytic combinatorics [15, § VII.9]. We limit ourselves to a sketchy (albeit technical) discussion.

Recall that a finite singular point z_0 of a linear differential equation with analytic coefficients is called *regular* when all solutions $y(z)$ have moderate growth $y(z) = 1/(z - z_0)^{O(1)}$ as $z \rightarrow z_0$ in a sector with vertex at z_0 , or equivalently when the equation satisfies a formal property called Fuchs’ criterion. The solutions in the neighborhood of z_0 then have a simple structure: for some $t \in \mathbb{N}$, there exist linearly independent formal solutions of the form (with $z_0 = 0$)

$$y(z) = z^\lambda \sum_{k=0}^{t-1} \phi_k(z) \log^k z, \quad \lambda \in \mathbb{C}, \quad \phi_k \in \mathbb{C}[[z]] \quad (12)$$

in number equal to the order r of the differential equation. Additionally, the ϕ_k converge on the open disk centered at 0 extending to the nearest singular point, so that the solutions (12) in fact form a basis of analytic solutions on any open sector with vertex at the origin contained in this disk. (See for instance [28] for proofs and references.)

Several extensions of the method of indeterminate coefficients used to obtain power series solutions at ordinary points allow to determine the coefficients of the series ϕ_k . We proceed to revisit Poole’s variant [28, §16] of Heffter’s method [22, Kap. 8] using the “operator algebra point of view” on indeterminate coefficients: a recursive formula for the coefficients of the series expansion of a solution is obtained by applying simple rewriting rules to the equation. This formulation makes the algorithm simpler for our purposes (compare [31, 11, 33]) and leads to a small complexity improvement. It also proves helpful in error control (§6).

Since our interest lies in the D-finite case, we assume that 0 is a regular singular point of Equation (1). Letting $\theta = z \frac{d}{dz}$, the equation rewrites as $L(z, \theta) \cdot y = 0$ where L is a polynomial in two noncommutative indeterminates (and $L(z, \theta)$ has no nontrivial left divisor in $K[z]$). Based on (12), we seek solutions as *formal logarithmic sums*

$$y(z) = \sum_{n \in \lambda + \mathbb{Z}} \sum_{k \geq 0} y_{n,k} \frac{\log^k z}{k!} z^n, \quad \lambda \in \mathbb{C}.$$

Let us call the double sequence $\mathbf{y} = (y_{n,k})_{n \in \lambda + \mathbb{Z}, k \in \mathbb{N}}$ the *coefficient sequence* of y . The *shift operators* S_n and S_k on such double sequences are defined by $S_n \cdot \mathbf{y} = (y_{n+1,k})$, and $S_k \cdot \mathbf{y} = (y_{n,k+1})$. The heart of the method lies in the following observation.

PROPOSITION 3. *Let $y(z)$ be a formal logarithmic sum. The relation $L(z, \theta) \cdot y = 0$ holds (formally) iff the coefficient sequence $\mathbf{y}$ of y satisfies $L(S_n^{-1}, n + S_k) \cdot \mathbf{y} = 0$.*

PROOF. The operators z and θ act on logarithmic sums by $z \cdot y(z) = \sum_{n \in \lambda + \mathbb{Z}} \sum_{k \geq 0} y_{n-1,k} \frac{\log^k z}{k!} z^n$ and $\theta \cdot y(z) = \sum_{n \in \lambda + \mathbb{Z}} \sum_{k \geq 0} (ny_{n,k} + y_{n,k+1}) \frac{\log^k z}{k!} z^n$. Thus the coefficient sequence of $L(z, \theta) \cdot y$ is $L(S_n^{-1}, n + S_k) \cdot \mathbf{y}$. $\square$

Assume that $y(z)$ satisfies (1). Then $L(S_n^{-1}, n + S_k) \cdot \mathbf{y} = 0$; additionally, (12) implies that $y_{n,k} = 0$ for $k \geq t$, which translates into $S_k^t \cdot \mathbf{y} = 0$. Set $L(z, \theta) = Q_0(\theta) + R(z, \theta)z$, and fix $n \in \lambda + \mathbb{Z}$. If $Q_0(n) \neq 0$, then $Q_0(n + X)$ is invertible in $K(\lambda)[[X]]$, and

$$\mathbf{y} = -(Q_0(n + S_k)^{-1} \bmod S_k^t) R(S_n^{-1}, n + S_k) S_n^{-1} \cdot \mathbf{y}.$$

In general, let $\mu(n)$ be the multiplicity of n as a root of Q_0 . Take $T_n \in K(\lambda)[X]$ such that $T_n(X)(X^{-\mu(n)} Q_0(n + X)) = 1 + O(X^t)$ (explicitly, $T_n(S_k) = \sum_{v=0}^{t-1} \left(\frac{\partial^v}{\partial X^v} \frac{X^{\mu(n)}}{Q_0(X)} \right)_{X=n} S_k^v$). Then, it holds that

$$S_k^{\mu(n)} \cdot \mathbf{y} = -T_n(S_k) R(S_n^{-1}, n + S_k) S_n^{-1} \cdot \mathbf{y}, \quad (13)$$

hence the $y_{n',k}$ with $n' < n$ determine $(y_{n,k})_{k \geq \mu(n)}$, while the $y_{n,k}$, $k < \mu(n)$ remain free.

Coupled with the condition $y_{n,k} = 0$ for $n - \lambda < 0$ following from (12), this implies that a solution y is characterized by $(y_{n,k})_{(n,k) \in E}$, where $E = \{(n,k) \mid k < \mu(n)\}$. As in §3, this choice of initial values induces that of canonical solutions (at 0) $y[n, k]$ defined by $y[n, k]_{n,k} = 1$ and $y[n, k]_{n',k'} = 0$ for $(n', k') \in E \setminus \{(n, k)\}$. The notion of transition matrix extends, see [33, §4] for a detailed discussion.

Equation (13) is a “recurrence relation” that lends itself to binary splitting. The main difference with the setting of §2 is that the companion matrix of the “recurrence operator” now contains truncated power series, *i.e.*, $B \in K(n)[[S_k]]/(S_k^t)$. The coefficients $y[u, v]_n = \sum_{k=0}^{t-1} y[u, v]_{n,k} \log^k z$ of canonical solutions may be computed fast by forming the product $B(n-1) \cdots B(u) \in K(\lambda)[[S_k]]/(S_k^t)$ and applying it to the initial values $Y_u = (0, \dots, 0, \log^v z)^T$. Compared to the algorithm of [33], multiplications of power series truncated to the order t replace multiplications of $t \times t$ submatrices, so that our method is slightly faster. As in §3, all entries of $M_{z_0 \rightarrow z_1}$ may (and should) be computed at once.

5. ERROR CONTROL

Performing numerical analytic continuation rigorously requires a number of bounds that control the behaviour of the

function under consideration. We now describe how error control is done in NumGfun. Some of the ideas used in this section appear scattered in [32]–[36]. NumGfun relies almost exclusively on *a priori* bounds; see [27, §5.2] for pointers to alternative approaches, and [36] for further useful techniques, including how to refine rough *a priori* bounds into better *a posteriori* bounds.

We start by discussing the computation of *majorant series* for the canonical solutions of the differential equation. Then we describe how these are used to determine, on the one hand, at which precision each transition matrix should be computed for the final result to be correct, and on the other hand, where to truncate the series expansions to achieve this precision. Finally, we propose a way to limit the cost of computing bounds in “bit burst” phases.

Remark 2. In practice, numerical errors that happen while computing the error bounds themselves are not always controlled, due to limited support for interval arithmetic in Maple. However, we have taken some care to ensure that crucial steps rely on rigorous methods.

Majorant series. A formal power series $g \in \mathbb{R}_+[[z]]$ is a *majorant series* of $y \in \mathbb{C}[[z]]$, which we denote by $y \leq g$, if $|y_n| \leq g_n$ for all n . If $y(z) \leq g(z)$ and $\hat{y}(z) \leq \hat{g}(z)$, then

$$\begin{aligned} y(z) &\leq g(|z|), & \frac{d}{dz} y(z) &\leq \frac{d}{dz} g(z), \\ y + \hat{y} &\leq g + \hat{g}, & y\hat{y} &\leq g\hat{g}, & y \circ \hat{y} &\leq g \circ \hat{g}. \end{aligned} \quad (14)$$

We shall allow majorant series to be formal logarithmic sums or matrices. The relation $\leq$ extends in a natural way: we write $\sum_{n,k} y_{n,k} z^n \log^k z \leq \sum_{n,k} g_{n,k} z^n \log^k z$ iff $|y_{n,k}| \leq g_{n,k}$ for all n and k , and $Y = (y_{i,j}) \leq G = (g_{i,j})$ iff Y and G have the same format and $y_{i,j} \leq g_{i,j}$ for all i, j . In particular, for scalar matrices, $Y \leq G$ if $|y_{i,j}| \leq g_{i,j}$ for all i, j . The inequalities (14) still hold.

Bounds on canonical solutions. The core of the error control is a function that computes $g(z)$ such that

$$\forall j, \quad y[z_0, j](z_0 + z) \leq g(z) \quad (15)$$

(in the notations of §3) given Equation (1) and a point z_0 . This function is called at each step of analytic continuation.

The algorithm is based on that of [27] (“SB” in what follows), designed to compute “asymptotically tight” *symbolic* bounds. Run over $\mathbb{Q}(i)$ instead of $\mathbb{Q}$, SB returns (in the case of convergent series) a power series satisfying (15) of the form $g(z) = \sum_{n=0}^{\infty} n!^{-\tau} \alpha^n \phi(n) z^n$, where $\tau \in \mathbb{Q}_+$, $\alpha \in \mathbb{Q}$, and $\phi(n) = e^{o(n)}$. The series g is specified by τ , α , and other parameters of no interest here that define ϕ . The tightness property means that τ and α are the best possible.

However, the setting of numerical evaluation differs from that of symbolic bounds in several respects: (i) the issue is no more to obtain human-readable formulae, but bounds that are easy to evaluate; (ii) bound computations should be fast; (iii) at regular singular points, the fundamental solutions are logarithmic sums, not power series. For space reasons, we only sketch the differences between SB and the variant we use for error control.

First, we take advantage of (i) to solve (ii). The most important change is that the parameter α is replaced by an approximation $\tilde{\alpha} \geq \alpha$. This avoids computations with algebraic numbers, the bottleneck of SB. Strictly speaking, it also means that we are giving up the tightness of the bound. However, in contrast with the “plain” majorant

series method [33, 34, 36], we are free to take $\tilde{\alpha}$ arbitrarily close to α , since we do not use the ratio $(\alpha/\tilde{\alpha})^n$ to mask subexponential factors. The algorithms from [27] adapt without difficulty. Specifically, Algorithms 3 and 4 are modified to work with $\tilde{\alpha}$ instead of α . Equality tests between “dominant roots” (Line 9 of Algorithm 3, Line 5 of Algorithm 4) can now be done numerically and are hence much less expensive. As a consequence, the parameter T from §3.3 is also replaced by an upper bound. This affects only the parameter $\phi(n)$ of the bound, which is already pessimistic.

The issue (iii) can be dealt with too. One may compute a series g such that $y(z) \leq g(z) \sum_{k=0}^{t-1} \frac{\log^k z}{k!}$ (with the notations of §4) by modifying the choice of K in [27, §3.3] so that $\sum_{k=0}^{t-1} \left| \left(\frac{\partial^k}{\partial X^k} \frac{X^{\mu(n)}}{Q_0(X)} \right)_{X=n} \right| \leq K/n^r$, and replacing [27, Eq. (14)] by Eq. (13) above in the reasoning.

Global error control. We now consider the issue of controlling how the approximations at each step of analytic continuation accumulate. Starting with a user-specified error bound ϵ , we first set ϵ' so that an ϵ' -approximation $\tilde{r} \in \mathbb{Q}(i)$ of the exact result r rounds to $\diamond(\tilde{r}) \in \bigcup_{n \in \mathbb{N}} 10^{-n} \mathbb{Z}[i]$ with $|\diamond(\tilde{r}) - r| < \epsilon$. No other *rounding* error occur, since the rest of the computation is done on objects with coefficients in $\mathbb{Q}(i)$. However, we have to choose the precision at which to compute each factor of the product $\Pi = \tilde{R} \tilde{M}_{m-1} \cdots \tilde{M}_0 \tilde{I}$ (in `evaldiffeq`, and of similar products for other analytic continuation functions) so that $|\tilde{r} - r| < \epsilon'$.

As is usual for this kind of applications, we use the Frobenius norm, defined for any (not necessarily square) matrix by $\|(a_{i,j})\|_F = (\sum_{i,j} |a_{i,j}|^2)^{1/2}$. The Frobenius norm satisfies $\|AB\|_F \leq \|A\|_F \|B\|_F$ for any matrices A, B of compatible dimensions. If A is a square $r \times r$ matrix,

$$\|A\|_\infty \leq \|A\|_2 \leq \|A\|_F \leq r \|A\|_\infty \quad (16)$$

where $\|\cdot\|_2$ is the matrix norm induced by the Euclidean norm while $\|\cdot\|_\infty$ denotes the *entrywise* uniform norm. Most importantly, the computation of $\|A\|_F$ is numerically stable, and if A has entries in $\mathbb{Q}(i)$, it is easy to compute a good upper bound on $\|A\|_F$ in floating-point arithmetic.

We bound the total error ϵ_Π on Π using repeatedly the rule $\|\tilde{A}\tilde{B} - AB\|_F \leq \|\tilde{A}\|_F \|\tilde{B} - B\|_F + \|\tilde{A} - A\|_F \|B\|_F$. From there, we compute precisions ϵ_A such that having $\|\tilde{A} - A\|_F < \epsilon_A$ for each factor A of Π ensures that $\epsilon_\Pi < \epsilon'$. Upper bounds on the norms $\|\tilde{A}\|_F$ and $\|A\|_F$ appearing in the error estimate are computed either from an approximate value of A (usually $\tilde{A}$ itself) if one is available at the time, or from a matrix G such that $A \leq G$ given by (14, 15).

Local error control. Let us turn to the computation of individual transition matrices. We restrict to the case of ordinary points. Given a precision ϵ determined by the “global error control” step, our task is to choose N such that $\|\tilde{M} - M_{z_0 \rightarrow z_1}\|_F \leq \epsilon$ if $\tilde{M}$ is computed by truncating the entries of (8) to the order N . If g satisfies (15), it suffices that $g_N^{(i)}(|z_1 - z_0|) \leq \frac{\epsilon}{r}$ for all i , as can be seen from (14, 16). We find a suitable N by dichotomic search. For the family of majorant series g used in NumGfun, the $g_N^{(i)}(x)$ are not always easy to evaluate, so we bound them by expressions involving only elementary functions [27, §4.2]. The basic idea, related to the saddle-point method from asymptotic analysis, is that if $x \geq 0$, inequalities like $g_n(x) \leq (x/t_n)^n g(t_n) (1 - x/t_n)^{-1}$ are asymptotically tight for suitably chosen t_n .

Points of large bit-size. Computing the majorants (15) is expensive when the point z_0 has large height. This can be fixed by working with an approximate value c of z_0 to obtain a bound valid in a neighborhood of c that contains z_0 . This technique is useful mainly in “bit burst” phases (where, additionally, we can reuse the same c at each step).

Assume that $Y[c](c+z) \leq G(z)$ for some $c \approx z_0$. Since $Y[z_0](z_0+z) = Y[c](c+(z_0-c)+z) M_{c \rightarrow z_0}^{-1}$ by (7), it follows that $Y[z_0](z_0+z) \leq G(|z_0-c|+z) \|M_{c \rightarrow z_0}^{-1}\| \mathbf{1}$ where $\mathbf{1}$ is a square matrix filled with ones. Now $M_{c \rightarrow c+z} = \text{Id} + O(z)$, whence $M_{c \rightarrow c+z} - \text{Id} \leq G(z) - G(0)$. For $|z_0-c| < \eta$, this implies $\|M_{c \rightarrow z_0} - \text{Id}\|_F \leq \delta := \|G(\eta) - G(0)\|_F$. Choosing η small enough that $\delta < 1$, we get $\|M_{c \rightarrow z_0}^{-1}\|_F \leq (1-\delta)^{-1}$. If G was computed from (15), *i.e.*, for $G(z) = (\frac{1}{j!} g^{(j)}(z))_{i,j}$, this finally gives the bound

$$y[z_0, j](z_0+z) \leq \frac{r}{j! (1-\delta)} g^{(j)}(\eta+z),$$

valid for all z_0 in the disk $|z_0-c| < \eta$.

Note however that simple differential equations have solutions like $\exp(K/(1-z))$ with large K . The condition $\delta < 1$ then forces us to take c of size $\Omega(K)$. Our strategy to deal with this issue in NumGfun is to estimate K using a point c of size $O(1)$ and then to choose a more precise c (as a last resort, $c = z_0$) based on this value if necessary.

Remark 3. If the evaluation point z is given as a program, a similar reasoning allows to choose automatically an approximation $\tilde{z} \in \mathbb{Q}(i)$ of z such that $|y(\tilde{z}) - y(z)|$ is below a given error bound [32, §4.3]. In other applications, it is useful to have bounds on transition matrices that hold uniformly for *all* small enough steps in a given domain. Such bounds may be computed from a majorant differential equation with constant coefficients [35, §5].

6. REPEATED EVALUATIONS

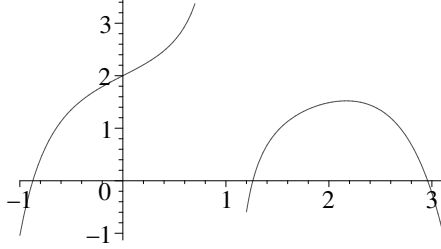
Drawing plots or computing integrals numerically requires to evaluate the same function at many points, often to a few digits of precision only. NumGfun provides limited support for this through the function `diffeqtoproc`, which takes as input a D-finite function y , a target precision ϵ , and a list of disks, and returns a Maple procedure that performs the numerical evaluation of y . For each disk $D = \{z, |z-c| < \rho\}$, `diffeqtoproc` computes a polynomial $p \in \mathbb{Q}(i)[z]$ such that $|\diamond(p(z)) - y(z)| < \epsilon$ for $z \in D \cap \mathbb{Q}(i)$, where $\diamond$ again denotes rounding to complex decimal. The procedure returned by `diffeqtoproc` uses the precomputed p when possible, and falls back on calling `evaldiffeq` otherwise.

The approximation polynomial p is constructed as a linear combination of truncated Taylor series of fundamental solutions $y[c, j]$, with coefficients obtained by numerical analytic continuation from 0 to c . The way we choose expansion orders is similar to the error control techniques of §5: we first compute a bound B_{can} on the fundamental solutions and their first derivatives on the disk D . The vector $Y(c)$ of “initial values” at c is computed to the precision ϵ'/B_{can} where $\epsilon' \leq \epsilon/(2r)$. We also compute $B_{\text{ini}} \geq \|Y(c)\|_F$. Each $y[c, j]$ is expanded to an order N such that $\|y[c, j]_N\|_{\infty, D} \leq \epsilon'/B_{\text{ini}}$, so that finally $|p(z) - y(z)| \leq \epsilon$ for $z \in D$.

The most important feature of `diffeqtoproc` is that it produces certified results. At low precisions and in the absence of singularities, we expect that interval-based numerical solvers will perform better while still providing (*a posteriori*) guarantees. Also note that our simple choice of p is far from

optimal. If approximations of smaller degree or height are required, a natural approach is to aim for a slightly smaller error $\|y - p\|_{\infty, D}$ above, and then replace p by a polynomial $\tilde{p}$ for which we can bound $\|p - \tilde{p}\|_{\infty}$ [36, §6.2].

Example 6. The following plot of the function y defined by $(z-1)y''' - z(2z-5)y'' - (4z-6)y' + z^2(z-1)y$ with the initial values $y(0) = 2$, $y'(0) = 1$, $y''(0) = 0$ was obtained using polynomial approximations on several disks that cover the domain of the plot while avoiding the pole $z = 1$. The whole computation takes about 9 s. Simple numerical integrators typically fail to evaluate y beyond $z = 1$.



7. FINAL REMARKS

Not all of NumGfun was described in this article. The symbolic bounds mentioned in §5 are also implemented, with functions that compute majorant series or other kinds of bounds on rational functions (`bound_ratpoly`), D-finite functions (`bound_diffeq` and `bound_diffeq_tail`) and P-recursive sequences (`bound_rec` and `bound_rec_tail`). This implementation was already presented in [27].

Current work focuses on adding support for evaluation “at regular singular points” (as outlined in §4), and improving performance. The development version of NumGfun already contains a second implementation of binary splitting, written in C and called from the Maple code. In the longer term, I plan to rewrite other parts of the package “from the bottom up”, both for efficiency reasons and to make useful subroutines independent of Maple.

Acknowledgements. I am grateful to my advisor, B. Salvy, for encouraging me to conduct this work and offering many useful comments. Thanks also to A. Benoit, F. Chyzak, P. Giorgi, J. van der Hoeven and A. Vaugon for stimulating conversations, bug reports, and/or comments on drafts of this article, and to the anonymous referees for helping make this paper more readable.

This research was supported in part by the MSR-INRIA joint research center.

8. REFERENCES

- [1] J. Abad, F. J. Gómez, and J. Sesma. An algorithm to obtain global solutions of the double confluent Heun equation. *Numer. Algorithms*, 49(1-4):33–51, 2008.
- [2] M. Beeler, R. W. Gosper, and R. Schroepel. Hakmem. AI Memo 239, MIT Artificial Intelligence Laboratory, 1972.
- [3] F. Bellard. Computation of 2700 billion decimal digits of Pi using a Desktop Computer. 2010. <http://bellard.org/pi/pi2700e9/>
- [4] D. J. Bernstein. Fast multiplication and its applications. In J. Buhler and P. Stevenhagen, editors, *Algorithmic Number Theory*, pages 325–384. Cambridge University Press, 2008.
- [5] A. Bostan, T. Cluzeau, and B. Salvy. Fast algorithms for polynomial solutions of linear differential equations. In *ISSAC'05*, pages 45–52. ACM press, 2005.
- [6] R. P. Brent. The complexity of multiple-precision arithmetic. In R. S. Anderssen and R. P. Brent, editors, *The Complexity of Computational Problem Solving*, pages 126–165, 1976.
- [7] R. P. Brent. A Fortran multiple-precision arithmetic package. *ACM Trans. Math. Softw.*, 4(1):57–70, 1978.
- [8] R. P. Brent and P. Zimmermann. *Modern Computer Arithmetic*. Version 0.4. 2009.
- [9] P. Birkisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*. Springer Verlag, 1997.
- [10] H. Cheng, G. Hanrot, E. Thomé, E. Zima, and P. Zimmermann. Time- and space-efficient evaluation of some hypergeometric constants. In *ISSAC'07*. ACM press, 2007.
- [11] D. V. Chudnovsky and G. V. Chudnovsky. On expansion of algebraic functions in power and Puiseux series (I, II). *J. Complexity*, 2(4):271–294, 1986; 3(1):1–25, 1987.
- [12] D. V. Chudnovsky and G. V. Chudnovsky. Computer algebra in the service of mathematical physics and number theory. In *Computers in mathematics (Stanford, CA, 1986)*, pages 109–232. 1990.
- [13] R. Dupont. *Moyenne arithmético-géométrique, suites de Borchardt et applications*. Thèse de doctorat, École polytechnique, Palaiseau, 2006.
- [14] P. Falloon, P. Abbott, and J. Wang. Theory and computation of spheroidal wavefunctions. *J. Phys. A*, 36:5477–5495, 2003.
- [15] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.
- [16] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.*, 33(2):13:1–13:15, 2007.
- [17] X. Gourdon and P. Sebah. Binary splitting method, 2001. <http://numbers.computation.free.fr/>
- [18] T. Granlund. *GMP*. <http://gmplib.org/>
- [19] B. Haible and R. B. Kreckel. *CLN*. <http://www.ginac.de/CLN/>
- [20] <http://ddmf.msr-inria.inria.fr/>
- [21] B. Haible and T. Papanikolaou. Fast multiprecision evaluation of series of rational numbers, 1997.
- [22] L. Heffter. *Einleitung in die Theorie der linearen Differentialgleichungen*. Teubner, Leipzig, 1894.
- [23] E. Jeandel. Évaluation rapide de fonctions hypergéométriques. Rapport technique RT-0242, INRIA-ENS Lyon, 2000.
- [24] O. M. Makarov. An algorithm for multiplying 3×3 matrices. *Comput. Math. Math. Phys.*, 26(1):179–180, 1987.
- [25] L. Meunier and B. Salvy. ESF: an automatically generated encyclopedia of special functions. In *ISSAC'03*, pages 199–206. ACM Press, 2003. <http://algo.inria.fr/esf/>
- [26] M. Mezzarobba. *Génération automatique de procédures numériques pour les fonctions D-finites*. Rapport de stage, Master parisien de recherche en informatique, 2007.
- [27] M. Mezzarobba and B. Salvy. Effective bounds for P-recursive sequences. *J. Symbolic Comput.*, to appear. arXiv:0904.2452v2.
- [28] E. G. C. Poole. *Introduction to the theory of linear differential equations*. The Clarendon Press, Oxford, 1936.
- [29] B. Salvy and P. Zimmermann. Gfun: A Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Trans. Math. Softw.*, 20(2):163–177, 1994. <http://algo.inria.fr/libraries/papers/gfun.html>
- [30] R. P. Stanley. *Enumerative combinatorics*, volume 2. Cambridge University Press, 1999.
- [31] É. Tournier. *Solutions formelles d'équations différentielles*. Thèse de doctorat, Université de Grenoble, 1987.
- [32] J. van der Hoeven. Fast evaluation of holonomic functions. *Theoret. Comput. Sci.*, 210(1):199–216, 1999.
- [33] J. van der Hoeven. Fast evaluation of holonomic functions near and in regular singularities. *J. Symbolic Comput.*, 31(6):717–743, 2001.
- [34] J. van der Hoeven. Majorants for formal power series. Technical Report 2003-15, Université Paris-Sud, 2003.
- [35] J. van der Hoeven. Efficient accelero-summation of holonomic functions. *J. Symbolic Comput.*, 42(4):389–428, 2007.
- [36] J. van der Hoeven. On effective analytic continuation. *Mathematics in Computer Science*, 1(1):111–175, 2007.
- [37] J. van der Hoeven. *Transseries et analyse complexe effective*. Mémoire d'habilitation, Université Paris-Sud, 2007.
- [38] A. Waksman. On Winograd's algorithm for inner products. *IEEE Trans. Comput.*, C-19(4):360–361, 1970.
- [39] P. Zimmermann. The bit-burst algorithm. Slides of a talk at the workshop “Computing by the Numbers: Algorithms, Precision, and Complexity” for R. Brent 60th birthday, 2006.

Chebyshev Interpolation Polynomial-Based Tools for Rigorous Computing

Nicolas Brisebarre and Mioara Joldes

LIP, Arénaire

CNRS/ENSL/INRIA/UCBL/Université de Lyon

46, allée d'Italie, 69364 Lyon Cedex 07, France

Nicolas.Brisebarre,Mioara.Joldes@ens-lyon.fr

ABSTRACT

Performing numerical computations, yet being able to provide rigorous mathematical statements about the obtained result, is required in many domains like global optimization, ODE solving or integration. Taylor models, which associate to a function a pair made of a Taylor approximation polynomial and a rigorous remainder bound, are a widely used rigorous computation tool. This approach benefits from the advantages of numerical methods, but also gives the ability to make reliable statements about the approximated function. Despite the fact that approximation polynomials based on interpolation at Chebyshev nodes offer a quasi-optimal approximation to a function, together with several other useful features, an analogous to Taylor models, based on such polynomials, has not been yet well-established in the field of validated numerics.

This paper presents a preliminary work for obtaining such interpolation polynomials together with validated interval bounds for approximating univariate functions. We propose two methods that make practical the use of this: one is based on a representation in Newton basis and the other uses Chebyshev polynomial basis. We compare the quality of the obtained remainders and the performance of the approaches to the ones provided by Taylor models.

Categories and Subject Descriptors

G.1 [Numerical Analysis]: Interval arithmetic, Multiple precision arithmetic, Interpolation, Approximation

General Terms

Algorithms, Reliability

Keywords

Rigorous Computing, Validated Numerics, Interpolation Polynomial, Chebyshev Polynomials, Taylor Models

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

1. INTRODUCTION

Computers are used nowadays to quickly give numerical solutions to various global optimization, ODE solving or integration problems. However, traditional numeric methods usually provide only approximate values for the solution. Bounds for the approximation errors are only sometimes available, are not guaranteed to be accurate or are sometimes unreliable. In contrast, validated computing aims at providing rigorously verified information about solutions, in order to complete proofs, or to give rigorous mathematical statements about the obtained result.

Interval arithmetic [21] is a classical tool to perform validated computations with floating-point arithmetic. Intervals are well-suited to represent enclosures of real numbers on a machine. However, they propagate only information about function values, and fail to convey much information about the other properties about the function itself. In particular, when modeling functions with interval arithmetic, splitting the domain in subintervals is usually required. For some cases, known as "high dependency problems" [5, 6, 12, 11] the number of necessary subintervals becomes unfeasible.

Taylor models [19, 5, 6], introduced by Berz and his group offer a remedy to this problem. They provide another way to rigorously manipulate and evaluate functions using floating-point arithmetic. They have been widely used for validated computing for global optimization and range bounding [18, 5, 11, 6], solutions of ODEs [23], quadrature [4], etc.

A Taylor model (TM) of order n for a function f which is supposed to be $n + 1$ times continuously differentiable over an interval $[a, b]$, is a rigorous polynomial approximation of f . More specifically, it is a couple (P, Δ) formed by a polynomial P of degree n , and an interval part Δ , such that $f(x) - P(x) \in \Delta, \forall x \in [a, b]$. Roughly speaking, the polynomial can be seen as a Taylor expansion of the function at a given point. The interval Δ provides the validation of the approximation, meaning that it provides an enclosure of all the approximation errors encountered (truncation, roundings).

A natural idea is to try to replace Taylor polynomials with better approximations such as minimax approximation, Chebyshev truncated series or interpolation polynomials (also called approximate Chebyshev truncated series when the points under consideration are Chebyshev nodes) for instance. The last two kind of approximations are of particular relevance for replacing Taylor polynomials since the series they define converge on domains better shaped for various usual applications than Taylor expansions (see, for instance, Section 2.7 of [9] for a more detailed account). Moreover, we can take advantage of numerous powerful techniques for computing

these approximations. So far, the attempts for using these better approximations, in the context of rigorous computing, do not seem to have succeeded, see for example [19] for a comparison of existing techniques.

In this work we propose two approaches for computing models based on interpolation polynomials at Chebyshev nodes, what we call “Chebyshev interpolation models” (CM). The first method is based on Newton Basis and the second on Chebyshev polynomial basis. We believe that bringing a certified remainder to an approximate truncated Chebyshev series and providing effective tools for working with such models, opens the way to adapting to rigorous computing many numerical algorithms based on Chebyshev interpolation polynomials, for rigorous ODE solving, quadrature, etc.

The outline of the paper is the following. We first recall or prove various definitions and results required by our approaches in Section 2. In Section 3, we present in more details TMs and we discuss the use of better polynomial approximations. Then, we introduce the notion of “Chebyshev interpolation models” in Section 4. The CMs are implemented using multiple precision interval arithmetic in order to perform rigorous computing and yet, for the sake of clarity, we present their implementation in exact arithmetic in Section 5. We give some results and a comparison of our models with TMs in Section 6. We end with a brief conclusion and a mention of our future works on the subject.

2. SOME PRELIMINARY STATEMENTS ABOUT INTERPOLATION AND CHEBYSHEV POLYNOMIALS

We first give a very short reminder on Chebyshev polynomials. A detailed presentation can be found in [7, 29]. Then we state some interpolation results that we use in the sequel.

2.1 Some basic facts about Chebyshev polynomials

Over $[-1, 1]$, Chebyshev polynomials can be defined as $T_n(x) = \cos(n \arccos x)$, $n \geq 0$. Since we consider functions over any interval $I = [a, b]$, we define in the following the Chebyshev polynomials over I as $T_n^{[a,b]}(x) = T_n\left(\frac{2x-b-a}{b-a}\right)$.

$T_{n+1}^{[a,b]}$ has $n+1$ distinct real roots in $[a, b]$, called “Chebyshev nodes” since they are of utmost interest for interpolation:

$$x_i^* = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(i+1/2)\pi}{n+1}\right), \quad i = 0, \dots, n. \quad (1)$$

We now recall

LEMMA 2.1. *The polynomial $W_{x^*}(x) = \prod_{i=0}^n (x - x_i^*)$, is the monic degree- $n+1$ polynomial that minimizes the supremum norm over $[a, b]$ of all monic polynomials in $\mathbb{C}[x]$ of degree at most $n+1$. We have*

$$W_{x^*}(x) = \frac{(b-a)^{n+1}}{2^{n+1}} T_{n+1}^{[a,b]}(x)$$

and

$$\max_{x \in [a,b]} |W_{x^*}(x)| = \frac{(b-a)^{n+1}}{2^{n+1}}.$$

2.2 Brief overview of interpolation results used

Let $I = [a, b]$ be an interval. Let f be a function that is at least $n+1$ times continuously differentiable over I . Let $\{y_i, i = 0, \dots, n\}$ be a set of $n+1$ points in I . There exists a unique polynomial P of degree $\leq n$ which interpolates f at these points [10]: $P(y_i) = f(y_i)$, $i = 0, \dots, n$, or if y_i is repeated k times, $P^{(j)}(y_i) = f^{(j)}(y_i)$, $j = 0, \dots, k-1$. If all the points are distinct, this is called Lagrange interpolation. In the extreme case that all the y_i are equal, P is just the Taylor polynomial of f at the considered point.

Several algorithms and interpolation formulae in various basis exist for representing P , for example monomial basis, Lagrange, Newton, Barycentric Lagrange, Chebyshev basis [10, 3, 31]. The numerical properties (stability) of these formulas have been widely studied in the literature [17].

Let us consider the polynomial P in Newton basis: $P(x) = \sum_{i=0}^n c_i N_{\bar{y},i}(x)$, where $N_{\bar{y},0}(x) = 1$ and $N_{\bar{y},i} = \prod_{j=0}^{i-1} (x - y_j)$, $i = 1, \dots, n$. The coefficients c_i are the divided-differences $f[y_0, \dots, y_i]$ of f at the points $y_0, \dots, y_i$. As mentioned above, if k points coincide, it suffices to take the successive $k-1$ derivatives of f . Note that c_i can be obtained thanks to the divided-differences algorithm [15, 31]. Moreover, the error between f and P is given [15, 31] by:

$$\forall x \in I, f(x) - P(x) = f[y_0, \dots, y_n, x] W_{\bar{y}}(x) \quad (2)$$

with $W_{\bar{y}}(x) = \prod_{i=0}^n (x - y_i)$. By a repeated application of Rolle’s theorem [10, 15, 31], we get: $\forall x \in I$, $\exists \xi \in (a, b)$ s.t.

$$f(x) - P(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) W_{\bar{y}}(x). \quad (3)$$

In the sequel we denote the right member of this formula for the error by $\Delta_n(x, \xi)$. Lemma 2.1 suggests that a clever choice of interpolation points seems to be the Chebyshev nodes (1), which is indeed the case as mentioned in 3.2.2.

We will take advantage of the following lemma which generalizes Lemma 5.12 of [33]:

LEMMA 2.2. *Under the assumptions on f and y_i above, if $f^{(n+1)}$ is increasing (resp. decreasing) over I , then $f[y_0, \dots, y_n, x]$ is increasing (resp. decreasing) over I .*

PROOF. Assume that $f^{(n+1)}$ is increasing. We know (see Chap. 4 of [31] for instance) that, for all $x \in [a, b]$, we have

$$f[y_0, \dots, y_n, x] = \int_0^1 \int_0^{t_1} \dots \int_0^{t_n} f^{(n+1)}(y_0 + t_1(y_1 - y_0) + \dots + t_{n+1}(x - y_n)) dt_1 \dots dt_{n+1}.$$

Let $x, y \in [a, b]$, $x \leq y$, let Z_n denote $y_0 + t_1(y_1 - y_0) + \dots + t_n(y_n - y_{n-1}) - t_{n+1}y_n$, we notice that

$$f[y_0, \dots, y_n, y] - f[y_0, \dots, y_n, x] = \int_0^1 \int_0^{t_1} \dots \int_0^{t_n} (f^{(n+1)}(Z_n + t_{n+1}y) - f^{(n+1)}(Z_n + t_{n+1}x)) dt_1 \dots dt_{n+1}.$$

Since $t_{n+1} \geq 0$, we have $Z_n + t_{n+1}y \geq Z_n + t_{n+1}x$. As $f^{(n+1)}$ is increasing, it follows that the integrand is nonnegative, which implies $f[y_0, \dots, y_n, y] \geq f[y_0, \dots, y_n, x]$. $\square$

3. TAYLOR MODELS VS. USING BETTER APPROXIMATIONS

3.1 Basic principles of Taylor models

Computing a TM consists in computing a polynomial together with an interval bound by applying simple rules recursively on the structure of the function f . In fact, for functions like trigonometric, exponential, logarithmic functions, as well as operations like $1/x$ or the power function, all referred to in this article as basic functions (or as intrinsics in [18]), bounds for the remainders can be easily computed. For composite functions, TMs offer usually a much tighter bound than the one directly computed for the whole function, for example using automatic differentiation [1, 26, 21]. A meaningful comparison for this phenomenon is provided in [11, 19]. Here we provide the reader with a quick overview of the situation. Let u be a basic function defined on I and v a basic function defined on J , an interval enclosing the image $u(I)$. Consider the Taylor expansion of a composite function $v \circ u$ about $x_0 \in I$. The Taylor remainder is given by the Lagrange formula: for all $x \in I$, there exists $\xi \in (a, b)$ such that

$$R(x) = \frac{(v \circ u)^{(n+1)}(\xi)}{(n+1)!} (x - x_0)^{n+1}.$$

When bounding the remainder by means of automatic differentiation, an interval K enclosing the values of the $n+1$ -th derivative of this composite function $(v \circ u)^{(n+1)}(I)$ is obtained by performing many recursive operations involving enclosures of $u^{(i)}(I)$ and $v^{(i)}(J)$ which finally may produce a considerable overestimation in the remainder [11].

In contrast, TMs compute polynomials and interval bounds directly for v and u , and then use algebraic rules for computing a TM for the composition $v \circ u$. Since both v and u are basic functions, evaluating with interval arithmetic their $n+1$ -th derivative can be done in a fast way using simple formulae and does not lead to serious overestimation.

Moreover, most of the functions we deal with have Taylor series whose coefficients decrease. In particular, when the functions are analytic over a sufficiently large domain (a disk of radius > 1 suffices), the magnitude of the coefficients of the underlying polynomial decreases exponentially (this is a consequence of Cauchy's integral formula). Hence when performing the composition of two such models, the intervals contributing to the final remainder become smaller for monomials of higher degree. This leads to a reduced overestimation in the computed remainder.

In conclusion, we highlight that, in practice, it is significantly more suitable to use a two-step procedure for handling composite functions: first step consists in computing models (P, Δ) for all basic functions; second, apply algebraic rules specifically designed for handling operations with these mixed models instead of operations with the corresponding functions.

3.2 Substituting tighter polynomial approximations to Taylor polynomials

It is well known that Taylor polynomials can be fairly poor approximations to functions, except perhaps on very small intervals. The idea of using better approximation polynomials, for example Chebyshev truncated series, in the context of validated computations was introduced in [14] under the name ultra-arithmetic, which we first briefly present. Then, we turn to a first introduction of the CMs and give the basic principles which they rely on. We finally address the case of minimax approximation. The case of truncated Chebyshev series is mentioned in the conclusion of this paper.

3.2.1 A previous attempt

As explained in [19], in the setting of ultra-arithmetic, the advantages of non-Taylor approximations cannot be explicitly maintained due to several drawbacks. It is noted in [19] that the Taylor representation is a special case, because for two functions f_1 and f_2 , the Taylor representation of order n , for the product $f_1 \cdot f_2$ can be obtained merely from the Taylor expansions of f_1 and f_2 , simply by multiplying the polynomials and discarding the orders $n+1$ to $2n$. On the contrary, the Chebyshev truncated series of a product $f_1 \cdot f_2$ can in general not be obtained from the Chebyshev representations of the factors f_1 and f_2 , and no operation analogous to TMs multiplication is given. Moreover, there is no systematic treatment of common basic functions. Finally, [19] explains that the methods developed in ultra-arithmetic can lead to an increase in the magnitude of the coefficients, which will increase both the computational errors and the difficulty of finding good interval enclosures of the polynomials involved.

3.2.2 Chebyshev interpolation polynomial

A natural idea is to consider an interpolation polynomial instead of a Taylor approximation: if the polynomial interpolates the function at Chebyshev nodes, such a polynomial, which we call Chebyshev interpolation polynomial, can usually provide a near-optimal approximation of f . See e.g. [25] which states an effective measure of this property.

Trefethen [32] uses the idea of representing the functions by Chebyshev interpolation polynomials. He chooses their expansion length in such a way as to maintain the accuracy of the approximation obtained close to machine precision. Moreover, the idea of using basic functions and then consider an algebra on them is used. The developed software, Chebfun was successfully used for numerically solving differential equations, quadrature problems. However, the Chebfun system does not provide any validation of the results obtained. As mentioned in [32], the aim of this system are numerical computations and no formal proof or safeguards are yet implemented to guarantee a validated computation, although it seems to be a future project.

3.2.3 Towards Chebyshev interpolation models

The interpolation polynomial itself is easy to compute and a wide variety of methods and various basis representations exist (we refer the reader to any numerical analysis book, see [31] for instance, for its computation). Another advantage of interpolation polynomials is that a closed formula, hence an explicit bound, for the remainder exists, cf. (2) and (3). Lemma 2.1 and Formula (3) induces that for a Chebyshev interpolation polynomial of degree n , roughly speaking, compared to a Taylor remainder, the interpolation error will be scaled down by a factor of 2^{-n} . For bounding the remainder, the only remaining difficulty is to bound the term $f^{(n+1)}(\xi)$ for $\xi \in I$. However, as briefly discussed above and in [11] the advantage of using directly this formula for the remainder can not be effectively maintained if the interval bound is obtained using automatic differentiation because of the overestimation. Hence in what follows we will try to adapt the "basic bricks" approach used by TMs to one using interpolation polynomials.

3.2.4 Minimax approximation

One might first try to directly use the minimax poly-

mial which is the polynomial which minimizes the supremum norm of the approximation error. In fact, such a polynomial can be obtained only through an iterative procedure, namely Remez algorithm, which can be considered too computationally expensive in our context. But the main drawback is linked to the computation of the remainder: either obtaining a certified remainder in such a procedure, raises a significant dependency problem as discussed in [12, 11], or the closed formula, due to Bernstein [13], does not bring any improvement over Formula (3).

4. CHEBYSHEV INTERPOLATION MODELS

In the following we consider a Chebyshev Interpolation model (CM) of degree n for a function f over an interval I , as a couple (P, Δ) , in the following sense: P is a polynomial of degree n which is “closely related” to the Chebyshev interpolation polynomial for f , Δ is an interval enclosure for the remainder $f - P$. Specifically, from the mathematical point of view, for basic functions f , this polynomial P coincides with the Chebyshev interpolation polynomial and the computation of Δ is detailed in 5.1.2. For composite functions, we will use some algebraic rules defined in Section 5 in a similar manner to the TMs arithmetic. We insist that our setting is multiple-precision interval arithmetic and all algorithms chosen are adapted for and implemented specifically for it. Since a thorough description of their implementation is tedious we indicate some implementation hints in 4.2. Before doing this, we explain why we chose to use Newton and Chebyshev bases for performing operations over CMs.

4.1 Choice of representation basis

When computing an interpolation polynomial P two key choices are: the interpolating nodes (which are Chebyshev nodes (1), in this paper), and the basis used for implementation.

In our case, however, a third essential requirement emerges: one has to be able to find suitable algebraic rules for operations with models (P, Δ) that will not lead to overestimation of the remainder in the resulted model. This implies that addition, multiplication or composition of models made out of polynomials P , together with an interval remainder bound, has to be an effective process not only in terms of performance and quality of the polynomial obtained, but also in terms of quality of remainder. We note, that in what concerns addition, the representation of P in any basis will suffice, since this is a linear operator.

For multiplication, as discussed in 3.2.1, it has not been obvious so far how to devise an efficient algorithm that, given two functions f_1 and f_2 and their models (P_1, Δ_1) and (P_2, Δ_2) , where the polynomial part is obtained through an interpolation process, is able to efficiently compute a model (P, Δ) for $f_1 \cdot f_2$. Furthermore, composition rules $f_1 \circ f_2$ were even more difficult to obtain. On the contrary, these operations are straightforward for TMs, which is one of their incontestable advantages.

Consequently, we were led to searching for suitable basis representations for P such that all these requirements are successful. We eliminated a Lagrange basis representation for P not only because of its poor numerical properties [17], but also because of the $n-1$ terms containing each $n-1$ products of $x - x_i^*$. This leads to high overestimations on the interval remainder when a composition operation was tried.

In what concerns using barycentric Lagrange basis versus Newton basis, it is proved in [17] that the first has better numerical properties. However, Newton basis also has good numerical properties for certain orderings of points [17]. In our case, barycentric Lagrange has the disadvantage that the polynomial P is represented as

$$\left(\sum_{i=0}^n \frac{w_i}{x - x_i^*} f(x_i^*) \right) / \left(\sum_{i=0}^n \frac{w_i}{x - x_i^*} \right).$$

It seems cumbersome to implement multiplication and composition of two models using this representation for P . We note that when composition should be implemented, the reciprocal $1/(x - x_i^*)$ should be designed. This leads to considerable overestimations of the remainder.

We note that Newton basis can be seen as an “immediate extension” to Taylor basis, so it is a natural try to make and leads to an adaptation of TMs algorithms. In what follows we consider the Newton basis at Chebyshev nodes taken in decreasing order, as given by Formula (1). This order has good numerical properties, see for example Section 5.3 [16]. In Section 5 we show how multiplications and composition operations can be implemented.

As for Chebyshev basis, we were led to this choice mainly because of the remarkable properties and success of numerical methods based on Chebyshev series expansions [9, 3].

Moreover, as we will see in Section 5.4, for most of the functions we consider, the rapid decreasing of the coefficients in the representations in the last two bases allow for small overestimation when handling CMs.

We now give some basic operations that will be used during the operations on the models.

Operations with polynomials in Newton basis.

Consider two polynomials of degree n in Newton basis:

$$P(x) = \sum_{i=0}^n p_i N_{x^*,i}(x) \text{ and } Q(x) = \sum_{i=0}^n q_i N_{x^*,i}(x).$$

Addition. Adding two polynomials in Newton basis is straightforward: $P(x) + Q(x) = \sum_{i=0}^n (p_i + q_i) N_{x^*,i}(x)$

Multiplication. When multiplying two polynomials in Newton basis, we need that “the lower part” of the result be represented in this basis also, and the “upper part”, should be represented such that it can be easily bounded with interval arithmetic. Hence, we chose to represent the product PQ in the basis $N_{x^*,0}, \dots, N_{x^*,n}, W_{x^*} N_{x^*,0}, \dots, W_{x^*} N_{x^*,n-1}$. We have $P(x) \cdot Q(x) = G(x) + W_{x^*}(x)H(x)$, where $G(x) = \sum_{i=0}^n g_i N_{x^*,i}(x)$ and $H(x) = \sum_{i=0}^{n-1} h_i N_{x^*,i}(x)$.

One of the advantages of this representation is that we gave an exact interval bound for W_{x^*} in Section 2.1. Moreover, it can be easily shown that $G(x)$ is the Chebyshev interpolation polynomial of $P(x) \cdot Q(x)$, i.e. of fg .

In order to compute the coefficients of G and H one can use several techniques. We mention that based on [8] a conversion back to monomial basis, followed by an interpolation seems to be the fastest asymptotically (needing $O(M(n) \log n)$ operations, where $M(n)$ denotes the cost of multiplying univariate polynomials of degree less than n). However, in our current multiple precision interval arithmetic implementation, we use an $O(n^2)$ algorithm based on applying the divided differences algorithm first for computing the coefficients of G and then for computing the coefficients of H =

$(PQ - G)/W_{x^*}$.

Interval Range Bounding. We use a Horner-like algorithm [16] for bounding the range of a polynomial in Newton basis, which takes $O(n)$ operations.

Operations with polynomials in Chebyshev basis.

Consider two polynomials of degree n in Chebyshev basis:

$$P(x) = \sum_{i=0}^n p_i T_i^{[a,b]}(x) \text{ and } Q(x) = \sum_{i=0}^n q_i T_i^{[a,b]}(x).$$

Addition. Adding two polynomials in Chebyshev basis is straightforward: $P(x) + Q(x) = \sum_{i=0}^n (p_i + q_i) T_i^{[a,b]}(x)$ and takes $O(n)$ operations.

Multiplication. Their product can be expressed in Chebyshev basis as follows: $P(x) \cdot Q(x) = \sum_{k=0}^{2n} c_k T_k^{[a,b]}(x)$, where $c_k = (\sum_{|i-j|=k} p_i \cdot q_j + \sum_{i+j=k} p_i \cdot q_j)/2$. This identity can be obtained noting [20] that $T_i^{[a,b]}(x) \cdot T_j^{[a,b]}(x) = (T_{i+j}^{[a,b]} + T_{|i-j|}^{[a,b]})/2$. The cost using this simple identity is $O(n^2)$ operations.

Interval Range Bounding. For our purpose, we will use the following identity for interval bounding range of a polynomial in Chebyshev basis, which takes $O(n)$ operations: $\forall x \in [a, b], P(x) \in p_0 + \sum_{i=1}^n p_i \cdot [-1, 1]$.

4.2 Rigorous implementation of the interpolation process

In the following, we denote an interval x as a pair $x = [\underline{x}, \overline{x}]$. For a polynomial P with real number coefficients, we denote by $\mathbf{P}$ a polynomial obtained by replacing its coefficients with intervals which enclose them.

We implemented all the operations involved using multiple-precision interval arithmetic and all the algorithms we use are straightforwardly adaptable in this context. Hence, from the implementation point of view, the only notable change is that instead of polynomials with real number coefficients we have polynomials with tight interval coefficients. This means that in our implementation, for a function f we obtain a model $(\mathbf{P}, \Delta)$, such that $\forall x \in [a, b], f(x) \in \mathbf{P}(x) + \Delta$. This design choice allows us to take into account all the rounding errors, because for each operation with interval arithmetic an outward rounding is performed. Moreover, it is proved in [28] that when evaluating a function φ over a point interval $x = [x, x]$, the interval enclosure of $\varphi(x)$ can be made arbitrarily tight by increasing the precision used for evaluation. In our case the computations needed for the coefficients of the polynomial are done with initially almost point intervals, so the overestimation of the coefficients can be made as small as possible by increasing the precision used.

However, if needed, a certified floating-point approximation polynomial can be easily obtained from a CM $(\mathbf{P}, \Delta)$.

Let us consider $\mathbf{P}(x) = \sum_{i=0}^n c_i \beta_i(x)$ where β_i is the i th element of the considered basis (Newton or Chebyshev). It suffices to take the middle points t_i of c_i as the coefficients of the approximation polynomial $\tilde{P}(x) = \sum_{i=0}^n t_i \beta_i(x)$ and then

to compute a simple interval bound $\delta = \sum_{i=0}^n [c_i - t_i, \overline{c_i} - t_i] \cdot \beta_i(x)$, using the methods presented in Section 4.1. Then the error between the function f and its approximation poly-

nomial $\tilde{P}$ is bounded by $\delta + \Delta$.

5. CHEBYSHEV INTERPOLATION MODELS WITH EXACT ARITHMETIC

We follow the two-step approach specific for TMs, as mentioned in Section 3. As detailed below, firstly, we compute models for basic functions; secondly we apply specifically designed algebraic operations on such models.

5.1 Basic functions

We consider as basic functions the trigonometric, exponential, logarithmic functions, $1/x$, the power function, or any other function for which specific properties can be exploited. For such functions we compute directly a model (P, Δ) formed by an interpolation polynomial P and an interval bound for the remainder Δ .

5.1.1 Computation of the interpolation polynomial

We mentioned in 2.2 how to express an interpolation polynomial in Newton basis, using the divided differences procedure. The number of operations necessary to compute the coefficients of the interpolation polynomial is $3n^2/2$ [16].

We now need to know how to represent it using Chebyshev basis. The Chebyshev interpolation polynomial P can be expressed using the collocation method [20] as follows:

$$P(x) = \sum_{i=0}^n p_i T_i^{[a,b]}(x), \text{ with } p_i = \sum_{k=0}^n \frac{2}{n+1} f(x_k^*) T_i^{[a,b]}(x_k^*).$$

Using directly this formula, the computation cost is $O(n^2)$ operations. It is known [24] that the usage of Fast Fourier Transform, can speed-up this computation to $O(n \log n)$ operations. However, note that in this case, an interval arithmetic adaptation of FFT should be considered.

5.1.2 Computation of the remainder

We can compute an enclosure of the remainder $f - P$ over $[a, b]$ using Formula (3): $\Delta_n(x, \xi) = f^{(n+1)}(\xi) W_{x^*}(x)/(n+1)!$, where $x, \xi \in [a, b]$. This reduces to bounding $f^{(n+1)}$ over $[a, b]$, which does not pose any problem for basic functions, since simple formulae are available for their derivatives. Moreover, $W_{x^*}(x)$ can be bounded straightforwardly using Lemma 2.1.

We remark that, thanks to Lemma 2.2, an exact bound for the remainder can be computed when one can show that $f^{(n+1)}$ is increasing (resp. decreasing) over $[a, b]$. In such cases, one can use (2) and bound $f[x_0^*, \dots, x_n^*, x]$ by $[f[x_0^*, \dots, x_n^*, a], f[x_0^*, \dots, x_n^*, b]]$ (resp. $[f[x_0^*, \dots, x_n^*, b], f[x_0^*, \dots, x_n^*, a]]$). For basic functions, checking whether $f^{(n+2)}$ has constant sign over $[a, b]$ is simple. This remark makes it possible to obtain smaller remainders and strengthens the effectiveness of the “basic bricks” approach.

5.1.3 Bounding the interpolation polynomial

When using this approach, one needs to compute bounds for the range of polynomials involved in such models. We denote by $B(P)$ an interval range bound for a polynomial P , over a given interval I . Several methods exist and a trade-off between their speed and the tightness of the bound is usually considered. For TMs, the fastest but “rough” method is a Horner-like interval evaluation. More complicated schemes exist, that usually give tighter bounds: LDB, QDB [5], or using a conversion to Bernstein basis [22, 33]. For univariate polynomials, slower techniques based on root isolation can

be used for a very tight polynomial bounding [30]. In our case, we focused on speed, and so, when considering interval range bounding for polynomials in Newton or Chebyshev basis we used two simple methods described in Section 4.1. Similarly to TMs, for a penalty in speed, more refined algorithms can also be plugged-in. However the simple techniques we used, proved effective in most of the examples we treated so far.

5.2 Addition and multiplication

In what follows we consider two CMs for two functions f_1 and f_2 , over I , of degree n : (P_1, Δ_1) and (P_2, Δ_2) . Adding the two models is done by adding the two polynomials and the remainder bounds:

$$(P_1, \Delta_1) + (P_2, \Delta_2) = (P_1 + P_2, \Delta_1 + \Delta_2).$$

Note that adding two polynomials in Newton or Chebyshev basis is straightforward and has a linear complexity, see Section 4.1.

For proving the correction we note that: $\forall x \in I, \exists \delta_1 \in \Delta_1$ and $\delta_2 \in \Delta_2$ s.t. $f_1(x) - P_1(x) = \delta_1$ and $f_2(x) - P_2(x) = \delta_2$. Hence $f_1(x) + f_2(x) - (P_1(x) + P_2(x)) = \delta_1 + \delta_2 \in \Delta_1 + \Delta_2$.

For multiplication, we have

$$f_1(x) \cdot f_2(x) = P_1(x) \cdot P_2(x) + P_2(x) \cdot \delta_1 + P_1(x) \cdot \delta_2 + \delta_1 \cdot \delta_2.$$

We observe that $P_1 \cdot P_2$ is a polynomial of degree $2n$. Depending on the basis used, we split it into two parts: the polynomial consisting of the terms that “do not exceed n ”, $(P_1 \cdot P_2)_{0 \dots n}$ and respectively the upper part $(P_1 \cdot P_2)_{n+1 \dots 2n}$, for the terms of the product $P_1 \cdot P_2$ whose “order exceeds n ”.

Now, a CM for $f_1 \cdot f_2$ can be obtained by finding an interval bound Δ for all the terms except $P = (P_1 \cdot P_2)_{0 \dots n}$:

$$\Delta = B((P_1 \cdot P_2)_{n+1 \dots 2n}) + B(P_2) \cdot \Delta_1 + B(P_1) \cdot \Delta_2 + \Delta_1 \cdot \Delta_2.$$

The interval bound for the polynomials involved can be computed as discussed in 5.1.3.

In our current setting, cf. Section 4.1, the number of operations necessary to multiply two such models is $O(n^2)$.

For the sake of completeness, we mention that multiplying a CM with a constant, reduces to multiplying the polynomial and the remainder with the respective constant.

5.3 Composition

When the model for $f_1 \circ f_2$ is needed, we can consider $(f_1 \circ f_2)(x)$ as function f_1 evaluated at point $y = f_2(x)$. Hence, we have to take into account the additional constraint that the image of f_2 has to be included in the definition range of f_1 . This can be checked by a simple interval bound computation of $B(P_2) + \Delta_2$. Then we have:

$$(f_1 \circ f_2)(x) - P_1(f_2(x)) \in \Delta_1 \quad (4)$$

which implies that

$$(f_1 \circ f_2)(x) - P_1(P_2(x) + \Delta_2) \in \Delta_1 \quad (5)$$

In this formula, the only polynomial coefficients and remainders involved are those of the CMs of f_1 and f_2 which are basic functions. As we have seen above, fairly simple formulae exist for computing the coefficients and remainders of such functions. However, when using Formula (4), it is not obvious how to extract a polynomial and a final remainder

bound from it. In fact, we have to reduce this extraction process to performing just multiplications and additions of CMs. A similar idea is used for composing TMs [19, 33].

In our case, the difference is that P_1 and P_2 are polynomials represented in Newton/Chebyshev basis, and not in the monomial basis. In consequence, for computing the composition, we had to use a different algorithm. It is worth mentioning that a simple change of basis back and forth to monomial basis will not be successful. The problem is that the multiplications and additions used in such a composition process do not have to add too much overestimation to the final remainder. As we discussed in Section 3, for Taylor expansions of most of the functions we address, the size of the coefficients for the representation in the monomial basis is bounded by a decreasing sequence. Hence the contributions of the remainders in such a recursive algorithm are smaller and smaller. On the contrary, for interpolation polynomials, the coefficients represented in monomial basis oscillate too much and have poor numerical properties. Hence, a direct application of the principle of composing TMs will not be successful.

When using Newton basis, we perform the composition using a Horner-like algorithm [16]. It takes a linear number of operations between models.

ALGORITHM 1. *Horner-like composition of CMs in Newton basis: Composing (P_1, Δ_1) with (P_2, Δ_2)*

```
/*We denote by  $P_{1j}$  the  $j$ th coefficient of  $P_1$ */
/*and by  $x_j^*$  the  $j$ th interpolation point for  $P_1$ */
( $C_n, \mathbf{R}_n$ ) := ( $P_{1n}, [0, 0]$ )
For  $j = n - 1, \dots, 0$  do
( $C_j, \mathbf{R}_j$ ) := (( $P_2, \Delta_2$ ) - ( $x_j^*, [0, 0]$ )) · ( $C_{j+1}, \mathbf{R}_{j+1}$ ) + ( $P_{1j}, [0, 0]$ );
Return ( $C_0, \mathbf{R}_0 + \Delta_1$ )
```

When using Chebyshev basis, we perform the composition using an adaptation of Clenshaw algorithm [20]. Algorithm 2 is used for efficient evaluation of a Chebyshev sum $\sum c_i T_i(x)$. It reduces evaluation of such polynomials to basic additions and multiplications and it is as efficient as Horner form for evaluating a polynomial as a sum of powers using nested multiplications. In our case, the variable x where the sum is to be evaluated is a CM, the multiplications and additions are operations between CMs. Moreover, using this algorithm, we perform a linear number of such operations between models.

ALGORITHM 2. *Clenshaw-like composition of CMs in Chebyshev basis: Composing (P_1, Δ_1) with (P_2, Δ_2)*

```
( $C_{n+2}, \mathbf{R}_{n+2}$ ) := (0, [0, 0]) /*CMs for 0*/
( $C_{n+1}, \mathbf{R}_{n+1}$ ) := (0, [0, 0])
/*We denote by  $P_{1j}$  the  $j$ th coefficient of  $P_1$ */
For  $j = n, \dots, 1$  do
( $C_j, \mathbf{R}_j$ ) :=  $2 \cdot (P_2, \Delta_2) \cdot (C_{j+1}, \mathbf{R}_{j+1}) - (C_{j+2}, \mathbf{R}_{j+2}) + (P_{1j}, [0, 0])$ ;
( $C, \mathbf{R}$ ) := ( $P_2, \Delta_2$ ) · ( $C_1, \mathbf{R}_1$ ) - ( $C_2, \mathbf{R}_2$ ) + ( $P_{10}, [0, 0]$ )
Return ( $C, \mathbf{R} + \Delta_1$ )
```

5.4 Growth of the coefficients and overestimation

The overestimation does not grow too much during the recursion process. This is due to the nice convergence properties of the series expansions in Newton Basis or in Chebyshev polynomial basis. One can prove for instance that if the function under consideration is analytic over a “sufficiently large”

domain of $\mathbb{C}$ which contains $[a, b]$, the coefficients of the expansion in Newton [15] or in Chebyshev [9] bases decrease exponentially. As with TMs, when composing two such models, the intervals contributing to the final remainder become smaller for higher coefficients, which yields a reduced overestimation in the final remainder.

6. EXPERIMENTAL RESULTS

We implemented the two methods in Maple, using the IntpakX¹ package. The Maple code can be downloaded from <http://www.ens-lyon.fr/LIP/Arenaire/Ware/ChebModels>. Table 1 shows the quality of some absolute error bounds obtained with the following presented methods: CMs, direct interpolation combined with AD, and TMs respectively. Each row of the table represents one example. The function f , the interval I and the degree n of the approximation polynomial are given in the first column.

We computed, as explained in Section 5, two CMs: one using polynomials in Newton basis $CM_1 = (P_1, \Delta_1)$ and the other $CM_2 = (P_2, \Delta_2)$ using Chebyshev basis. Let $\Delta_1 = [\alpha_1, \beta_1]$ and $\Delta_2 = [\alpha_2, \beta_2]$, we provide in the second and third columns rigorous upper-bounds for $\max(|\alpha_1|, |\beta_1|)$ and $\max(|\alpha_2|, |\beta_2|)$ respectively.

In order to observe the amount of overestimation we computed the exact error bounds $\varepsilon_1 = \sup_{x \in I} \{|f(x) - P_1(x)|\}$ and $\varepsilon_2 = \sup_{x \in I} \{|f(x) - P_2(x)|\}$ and we provide in the fourth column the minimum of the two: $\varepsilon_{CM} = \min\{\varepsilon_1, \varepsilon_2\}$.

We give in fifth and sixth columns the computed remainder bounds and the exact error obtained when an interpolating polynomial is directly used (directly means that the remainder is computed using (3) and automatic differentiation). Finally we present the computed remainder bound obtained using a TM and the corresponding exact error. The Taylor polynomial was developed in the middle of I and the necessary polynomials bounds were computed using a Horner scheme.

The first five examples were analyzed in [11] for comparing the bounds obtained with interpolation vs. TMs. There, it was observed that in some cases the overestimation in the interpolation remainder is so big, that we can not benefit from using such a polynomial. We used them in order to highlight that CMs do not have this drawback and the remainders obtained with our methods have better quality than the TMs in all situations.

The first example presents a basic function which is analytic on the whole complex plane. There is almost no overestimation in this case, whatever method we use. The second is also a basic function. It has singularities in the complex plane (in $\pi/2 + \mathbb{Z}\pi$), but the interval I is relatively far from these singularities. All the methods present a relatively small overestimation. The third example is the same function but over a larger interval. In these case, the singularities are closer and Taylor polynomials are not very good approximations. The fourth and fifth examples are composite functions on larger intervals. The overestimation in the interpolation method becomes very large, rendering this method useless, while it stays reasonable with TMs and CMs.

The following examples (6 – 8) are similar to some presented in [3]. There, the authors computed the minimax polynomials for these functions. Evidently, the polynomials ob-

tained with CMs have a higher approximation error than the minimax, however, it is important to notice that in these tricky cases the remainder bound obtained for the CMs stays fairly good and is much better than the one obtained from TMs.

Examples 8–9 present the case when the definition domain of the function is close to a singularity. As seen in these examples, when a direct interpolation process is used for a composite function, unfortunately, one can not apply Lemma 2.2 for bounding the remainder. Consequently, the bound obtained for the remainder is highly overestimated. However, when using the approach based on “basic bricks” both TMs and CMs benefit from it, yielding a much better remainder bound.

Example 10 deals with a function which is associated to the classical Runge phenomenon. Firstly, since the complex singularities of the function f defined by $f(x) = 1/(1+4x^2)$ are close to the definition interval I , the Taylor polynomial is not a good approximation. Then, the interpolation method gives unfeasible interval remainder bounds due to the overestimation of the $n+1$ th derivative of the function f . TMs and CM1 fail both from the same cause: when computing a model for $f = g \circ h$ one needs to compose the models for $g(y) = 1/y$ and $h(x) = 1 + 4x^2$. The model for h is simple to compute. However, one has to take into account that the model of g has to be computed over an interval range enclosure of the model of h . When such an enclosure is computed, using the presented simple methods for polynomial range bounding we have an overestimation of the image of h . This leads in fact to an interval that contains 0. Hence, the model for g which is not defined in 0 can not be computed. On the contrary, the overestimation in polynomial range bounding with the method of CM2 is smaller and we have a feasible remainder in this case.

We do not give specific timings, since for the moment our implementation is rather a “proof of concept” one. The algorithms necessary for TMs seem to be slightly simpler and hence more efficient than the ones needed by our approaches: we observed a factor between 2 and 3 of speed-up in favor of TMs, for cases when Taylor approximations are fairly good ones (tight intervals, analytic functions). However, we note that in some cases (when considering larger intervals or composed functions), in order to attain the same quality for the remainder, TMs need a much higher polynomial degree, more computation time, or they have to be applied over many subintervals, which favors the usage of CMs.

7. CONCLUSION AND FUTURE WORK

We introduced two approaches for computing “Chebyshev interpolation models”, a tool which is potentially useful in various rigorous computing applications. Currently, they always achieve smaller remainders than Taylor models but require nevertheless more computing time in some cases.

This work is preliminary, in two senses. First, we did not address in this paper the opportunity to use Chebyshev truncated series instead of Chebyshev interpolation polynomials. Actually, this approach is a work in progress and seems very promising since the quality of the remainder should remain at least as good as the one provided by CM and the complexity of basic bricks computations should be lowered. This issue of complexity has to be addressed if we want CMs to replace TMs in most of univariate applications. The techniques developed in [27, 2] could prove useful to achieve this goal. Secondly, we address the computation of such models

¹<http://www.math.uni-wuppertal.de/~xsc/software/intpakX/>

No	$f(x), I, n$	CMs		Interpolation		TMs	
		CM1 bound	CM2 bound	Interp. bound	Exact bound	TM bound	Exact bound
1	$\sin(x), [3, 4], 10$	$1.19 \cdot 10^{-14}$	$1.19 \cdot 10^{-14}$	$1.13 \cdot 10^{-14}$	$1.19 \cdot 10^{-14}$	$1.13 \cdot 10^{-14}$	$1.16 \cdot 10^{-11}$
2	$\arctan(x), [-0.25, 0.25], 15$	$7.89 \cdot 10^{-15}$	$7.89 \cdot 10^{-15}$	$7.95 \cdot 10^{-17}$	$7.89 \cdot 10^{-15}$	$7.95 \cdot 10^{-17}$	$3.24 \cdot 10^{-12}$
3	$\arctan(x), [-0.9, 0.9], 15$	$5.10 \cdot 10^{-3}$	$5.10 \cdot 10^{-3}$	$1.76 \cdot 10^{-8}$	$5.10 \cdot 10^{-3}$	$1.76 \cdot 10^{-8}$	$5.70 \cdot 10^{-3}$
4	$\exp(1/\cos(x)), [0, 1], 14$	$6.69 \cdot 10^{-7}$	$5.22 \cdot 10^{-7}$	$4.95 \cdot 10^{-7}$	0.11	$6.10 \cdot 10^{-7}$	$2.59 \cdot 10^{-3}$
5	$\frac{\exp(x)}{\log(2+x) \cos(x)}, [0, 1], 15$	$1.70 \cdot 10^{-8}$	$9.11 \cdot 10^{-9}$	$2.21 \cdot 10^{-9}$	0.18	$2.68 \cdot 10^{-9}$	$3.38 \cdot 10^{-5}$
6	$\sin(\exp(x)), [-1, 1], 10$	$4.10 \cdot 10^{-6}$	$9.47 \cdot 10^{-5}$	$3.72 \cdot 10^{-6}$	$4.42 \cdot 10^{-3}$	$3.72 \cdot 10^{-6}$	$1.55 \cdot 10^{-3}$
7	$\tanh(x + 0.5) - \tanh(x - 0.5), [-1, 1], 10$	$8.48 \cdot 10^{-3}$	$1.75 \cdot 10^{-3}$	$4.88 \cdot 10^{-7}$	$8.48 \cdot 10^{-3}$	$4.88 \cdot 10^{-7}$	$2.96 \cdot 10^{-3}$
8	$\sqrt{x + 1.0001}, [-1, 0], 10$	$3.64 \cdot 10^{-2}$	$3.64 \cdot 10^{-2}$	$3.64 \cdot 10^{-2}$	$3.64 \cdot 10^{-2}$	$3.64 \cdot 10^{-2}$	0.11
9	$\sqrt{x + 1.0001} \cdot \sin(x), [-1, 0], 10$	$3.10 \cdot 10^{-2}$	$3.32 \cdot 10^{-2}$	$3.08 \cdot 10^{-2}$	$3.21 \cdot 10^{33}$	$3.08 \cdot 10^{-2}$	$9.83 \cdot 10^{-2}$
10	$\frac{1}{1+4x^2}, [-1, 1], 10$	$+\infty$	$1.13 \cdot 10^{-2}$	$6.17 \cdot 10^{-3}$	$1.50 \cdot 10^7$	$4.95 \cdot 10^{-3}$	$8.20 \cdot 10^2$

Table 1: Examples of bounds obtained by several methods

for univariate functions only. A long-term project of ours is to extend these methods to the multivariate case.

The methods presented in this paper should be available in the months to come in the Sollya² tool.

Acknowledgements

The authors warmly thank the reviewers for their careful work. It made it possible to significantly improve the manuscript. They also would like to thank Martin Berz, Sylvain Chevillard, Kyoko Makino, Jean-Michel Muller and Markus Neher for many useful discussions.

8. REFERENCES

- [1] C. Bendsten and O. Stauning. TADIFF, a Flexible C++ Package for Automatic Differentiation Using Taylor Series. Technical Report 1997-x5-94, Technical Univ. of Denmark, April 1997.
- [2] A. Benoit and B. Salvy. Chebyshev expansions for solutions of linear differential equations. In J. May, editor, *ISSAC '09: Proceedings of the twenty-second international symposium on Symbolic and algebraic computation*, pages 23–30, 2009.
- [3] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Rev.*, 46(3):501–517, 2004.
- [4] M. Berz and K. Makino. New methods for high-dimensional verified quadrature. *Reliable Computing*, 5(1):13–22, 1999.
- [5] M. Berz and K. Makino. Rigorous global search using Taylor models. In *SNC '09: Proceedings of the 2009 conference on Symbolic numeric computation*, pages 11–20, New York, NY, USA, 2009. ACM.
- [6] M. Berz, K. Makino, and Y.-K. Kim. Long-term stability of the tevatron by verified global optimization. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 558(1):1 – 10, 2006. Proceedings of the 8th International Computational Accelerator Physics Conference - ICAP 2004.
- [7] P. Borwein and T. Erdélyi. *Polynomials and Polynomial Inequalities*. Graduate Texts in Mathematics, Vol. 161. Springer-Verlag, New York, NY, 1995.
- [8] A. Bostan and É. Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21(4):420–446, August 2005. Festschrift for the 70th Birthday of Arnold Schönhage.
- [9] J. P. Boyd. *Chebyshev and Fourier spectral methods*. Dover Publications Inc., Mineola, NY, second edition, 2001.
- [10] E. W. Cheney. *Introduction to Approximation Theory*. McGraw-Hill, 1966.
- [11] S. Chevillard, J. Harrison, M. Joldes, and C. Lauter. Efficient and accurate computation of upper bounds of approximation errors. 40 pages, RRLIP2010-2, 2010.
- [12] S. Chevillard, M. Joldes, and C. Lauter. Certified and fast computation of supremum norms of approximation errors. In *19th IEEE SYMPOSIUM on Computer Arithmetic*, pages 169–176, 2009.
- [13] D. Elliott, D. F. Paget, G. M. Phillips, and P. J. Taylor. Error of truncated Chebyshev series and other near minimax polynomial approximations. *J. Approx. Theory*, 50(1):49–57, 1987.
- [14] C. Epstein, W. Miranker, and T. Rivlin. Ultra-arithmetic i: Function data types. *Mathematics and Computers in Simulation*, 24(1):1 – 18, 1982.
- [15] A. O. Gel'fond. *Calculus of finite differences*. Hindustan Pub. Corp., Delhi, 1971. Translated from the Russian, International Monographs on Advanced Mathematics and Physics.
- [16] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2002.
- [17] N. J. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA J. Numer. Anal.*, 24(4):547–556, 2004.
- [18] K. Makino. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators*. PhD thesis, Michigan State University, East Lansing, Michigan, USA, 1998.
- [19] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 4(4):379–456, 2003.
- [20] J. C. Mason and D. C. Handscomb. *Chebyshev polynomials*. Chapman & Hall/CRC, Boca Raton, FL, 2003.
- [21] R. E. Moore. *Methods and Applications of Interval Analysis*. Society for Industrial Mathematics, 1979.
- [22] P. S. V. Nataraj and K. Kotecha. Global optimization with higher order inclusion function forms part 1: A combined taylor-bernstein form. *Reliable Computing*, 10(1):27–44, 2004.
- [23] M. Neher, K. R. Jackson, and N. S. Nedialkov. On Taylor model based integration of ODEs. *SIAM J. Numer. Anal.*, 45:236–262, 2007.
- [24] R. Platte and N. Trefethen. Chebfun: A new kind of numerical computing. Technical Report NA-08/13, Oxford University Computing Laboratory, Oct. 2008.
- [25] M. J. D. Powell. On the maximum errors of polynomial approximations defined by interpolation and by least squares criteria. *Comput. J.*, 9:404–407, 1967.
- [26] L. B. Rall. The arithmetic of differentiation. *Mathematics Magazine*, 59(5):275–282, 1986.
- [27] L. Rebillard. *Étude théorique et algorithmique des séries de Chebyshev solutions d'équations différentielles holonomes*. PhD thesis, Institut National Polytechnique de Grenoble, 1998.
- [28] N. Revol. Newton's algorithm using multiple precision interval arithmetic. *Numer. Algorithms*, 34(2):417–426, 2003.
- [29] T. J. Rivlin. *Chebyshev polynomials. From approximation theory to algebra*. Pure and Applied Mathematics. John Wiley & Sons, New York, 2nd edition, 1990.
- [30] F. Rouillier and P. Zimmermann. Efficient isolation of polynomial's real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2004.
- [31] M. Schatzman. *Numerical Analysis, A Mathematical Introduction*. Oxford University Press, 2002.
- [32] L. N. Trefethen. Computing numerically with functions instead of numbers. *Mathematics in Computer Science*, 1(1):9–19, 2007.
- [33] R. Zumkeller. *Global Optimization in Type Theory*. PhD thesis, École polytechnique, 2008.

²<http://sollya.gforge.inria.fr/>

Blind Image Deconvolution via Fast Approximate GCD*

Zijia Li
Key Laboratory of
Mathematics Mechanization
AMSS, Beijing 100190, China
lizijia@amss.ac.cn

Zhengfeng Yang
Shanghai Key Laboratory of
Trustworthy Computing
East China Normal University,
Shanghai 200062, China
zfyang@sei.ecnu.edu.cn

Lihong Zhi
Key Laboratory of
Mathematics Mechanization
AMSS, Beijing 100190, China
lzhi@mmrc.iss.ac.cn,
<http://www.mmrc.iss.ac.cn/~lzhi>

ABSTRACT

The problem of blind image deconvolution can be solved by computing approximate greatest common divisors (GCD) of polynomials. The bivariate polynomials corresponding to the z -transforms of several blurred images have an approximate GCD corresponding to the z -transform of the original image. Since blurring functions as cofactors have very low degree in general, this GCD will be of high degree. On the other hand, if we only have one blurred image and want to identify the original scene, the blurred image can be partitioned such that each part completely contains the blurring function, hence the blurring function becomes the GCD which is of low degree. Therefore, we design a specialized algorithm for computing GCDs of polynomials to recover true images in two different cases. The new algorithm is based on the fast GCD algorithm for univariate polynomials and the Fast Fourier Transform (FFT) algorithm. The complexity of our specialized algorithm for identifying both the true image and the blurring functions from blurred images of size $n \times n$ is $O(n^2 \log(n))$ in the case of blurring functions of very low degree. The algorithm has been implemented in Maple and can extract true images of hundreds by hundreds pixel images from blurred images in a few seconds.

Categories and Subject Descriptors

G.1.6 [Numerical Analysis]: Approximation; I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms; I.4.4 [Image Processing and Computer Vision]: Restoration

General Terms

Algorithms, Experimentation

Keywords

Blind image deconvolution, approximate GCD, Bezout ma-

*Zijia Li and Lihong Zhi are supported by the Chinese National Natural Science Foundation under Grants 60821002/F02, 60911130369 and 10871194. Zhengfeng Yang is supported by the Chinese National Natural Science Foundation under Grant 10901055 and Shanghai Natural Science Foundation under Grant 09ZR1408800.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

trix, Sylvester matrix, Fast Fourier Transform.

1. INTRODUCTION

A grey image can be represented by a matrix whose dimension is equal to the size of the image, and a color image includes three parameters (Red, Green, Blue), and can be represented by three matrices respectively. Suppose the original image matrix is P , the distorted and polluted image matrix is F . The blurring function matrix and the additive noise matrix of F are U and N respectively. Then we have the following relation between the true image matrix and the distorted image matrix:

$$F = P * U + N. \quad (1)$$

The measurement for noise in an image is the signal-to-noise ratio, or SNR. In [17], it is defined as:

$$\text{SNR} = 10 \log_{10} \left(\frac{\sigma_{P*U}^2}{\sigma_N^2} \right), \quad (2)$$

where σ_{P*U}^2 and σ_N^2 are the variances of the blurred image without noise and the additive noise respectively.

Blind image deconvolution is the process of identifying both the true image and the blurring function from the blurred images. Even if we assume the blurred picture is noise-free, which means N is a zero matrix or a matrix with very little entries (for example $\text{SNR} \geq 50$), it is still very difficult and expensive to obtain P and U from F . For example, the complexity of algorithms based on the approximate factorization of polynomials for blind deconvolution is $O(n^8)$ for an $n \times n$ image [16, 28]. However, if we know multiple blurred versions of the true image, or the blurred image can be partitioned such that each part completely contains the same blurring function, blind image deconvolution can be transformed to computing approximate GCDs of polynomials by using the z -transforms [35].

Definition 1. A two-dimensional z -transform maps the elements of an $m \times n$ matrix P to the coefficients of a bivariate polynomial $p(x, y)$:

$$p(x, y) = \mathbf{x}^T \cdot P \cdot \mathbf{y}, \quad (3)$$

where $\mathbf{x} = [1, x, x^2, \dots, x^{m-1}]^T$, $\mathbf{y} = [1, y, y^2, \dots, y^{n-1}]^T$.

By the two-dimensional z -transform, we map the elements of matrices to coefficients of bivariate polynomials. Hence, (1) is transferred into

$$f(x, y) = p(x, y) u(x, y) + n(x, y), \quad (4)$$

where f, p, u, n are z -transforms of F, P, U, N respectively. Such a model is applicable in all scenarios where the distortion can be modeled as a linear filter acting on the original image. For example, camera motion and intermediate medium in satellite photography can be modeled as (1) and (4) [7].

Suppose the original image matrix is P , matrices of two distorted images are F_1, F_2 . The blurring functions and additive noises of F_1, F_2 are U, V and N_1, N_2 respectively. So we have the following relations between original images and distorted images:

$$F_1 = P * U + N_1, \quad F_2 = P * V + N_2. \quad (5)$$

By the two-dimensional z -transform, (5) can be transferred into polynomial forms:

$$\begin{cases} f_1(x, y) = p(x, y)u(x, y) + n_1(x, y), \\ f_2(x, y) = p(x, y)v(x, y) + n_2(x, y), \end{cases} \quad (6)$$

where $f_1, p, u, n_1, f_2, v, n_2$ are z -transforms of $F_1, P, U, N_1, F_2, V, N_2$ respectively. We can compute the approximate GCD of bivariate polynomials $f_1(x, y)$ and $f_2(x, y)$ to reconstruct the true image. In practice, $\deg(u)$ and $\deg(v)$ are very low compared with $\deg(p)$. Hence, the approximate GCD $p(x, y)$ of $f_1(x, y)$ and $f_2(x, y)$ will be of high degree.

If there is only one available blurred RGB image, we assume that three channels have the same blurring function. As well, we can get the matrix form corresponding to each channel

$$F_1 = P_1 * U + N_1, \quad F_2 = P_2 * U + N_2, \quad F_3 = P_3 * U + N_3,$$

where F_1, F_2, F_3 are the blurring image matrices of each channel, P_1, P_2, P_3 are original image matrices respectively, U is the blurring filter matrix and N_1, N_2, N_3 are additive noises. By z -transforms, we have:

$$\begin{cases} f_1(x, y) = p_1(x, y)u(x, y) + n_1(x, y), \\ f_2(x, y) = p_2(x, y)u(x, y) + n_2(x, y), \\ f_3(x, y) = p_3(x, y)u(x, y) + n_3(x, y). \end{cases} \quad (7)$$

Hence the blurring function $u(x, y)$ is now the approximate GCD of f_1, f_2, f_3 which is of very low degree.

In [35], they described a fast algorithm for computing the approximate GCD of bivariate polynomials based on the Sylvester-type GCD algorithm for univariate polynomials and the Discrete Fourier Transform (DFT) algorithm. Their idea is to sample polynomials f_1 and f_2 in one variable at the DFT points on the unit circle:

$$x = e^{-\frac{2k\pi i}{m}}, \quad k = 0, 1, \dots, m-1,$$

and

$$y = e^{-\frac{2l\pi i}{n}}, \quad l = 0, 1, \dots, n-1,$$

where $\mathbf{i} = \sqrt{-1}$, m and n are the number of rows and columns of the matrix of a blurred image. The GCD of the resulting univariate polynomials is found by using the Sylvester-type GCD method. These GCD polynomials are again sampled at the DFT points, eventually generating two matrices that are scaled versions of the two dimensional DFT of the original image. They obtained an approximation of $p(x, y)$ by equalizing these two matrices and taking the two-dimensional inverse DFT. They assumed that DFT points are free of these points which will change the degree of the GCD or be the zeros of the GCD or the cofactors [35]. For

images of size $n \times n$, the complexity of their algorithm is $O(n^4)$, which is a substantial saving compared with the direct generalized Sylvester-type procedures for bivariate polynomials [14, 23, 42], which require about $O(n^6)$ operations.

The computation of the approximate GCD of univariate polynomials has been extensively studied with various approaches including the Euclidean method on the polynomial remainder sequence [4, 20, 32, 36], the singular value decomposition or QR decomposition of the Sylvester matrix [8, 9, 13], Padé approximation [34], iterative method [39] and other optimization strategies [25, 26, 27, 31]. Fast GCD algorithms for univariate polynomials based on displacement structure of the Sylvester matrix have also been proposed in [5, 29, 30, 41, 43].

It is well known that the Bezout matrix can also be used to compute GCDs of univariate polynomials [1, 2, 3, 5, 10, 11, 15, 38]. Compared with the Sylvester matrix, the Bezout matrix has smaller size. For example, the size of the Sylvester matrix of two polynomials with the same degree n is $2n \times 2n$, while the size of the Bezout matrix is $n \times n$. Moreover, since the degree of the blurring function is usually very low, the GCD polynomial $p(x, y)$ has high degree. We see below that, in this case (Example 1 and Example 2 in Section 4), it is more appealing to use the Bezout matrix. Because only a very small submatrix of the Bezout matrix is constructed and used for computing approximate GCDs.

On the other hand, although the Bezout matrix has smaller size, entries of the Bezout matrix are bilinear in the coefficients of polynomials. For polynomials of degree m , the cost to generate a full size Bezout matrix is already $O(m^2)$. Hence, when the degree of the cofactor is high, for instance, in recovering true images from one blurred grey or RGB images (Example 3 and Example 4 in Section 4), it becomes more efficient to use the Sylvester matrix to compute the GCD of univariate polynomials.

Section 2 is devoted to recall some notations and well-known facts about the Bezout matrix and the Sylvester matrix. We also present a fast algorithm for computing the approximate GCD of univariate polynomials. In Section 3, we describe an algorithm for computing the approximate GCD of bivariate polynomials based on the univariate GCD computation and FFT. We prove that the complexity of our new algorithm for blind image deconvolution of size $n \times n$ is $O(n^2 \log(n))$ in the case of blurring functions of very low degree. Experiments are done in Section 4. Some remarks for future work will be given in Section 5.

2. AN APPROXIMATE UNIVARIATE POLYNOMIAL GCD ALGORITHM

Suppose we are given two univariate polynomials $f_1, f_2 \in \mathbb{C}[x] \setminus \{0\}$ with $\deg(f_1) = m$ and $\deg(f_2) = n$, assume $m \geq n$,

$$\begin{cases} f_1 = u_m x^m + u_{m-1} x^{m-1} + \dots + u_1 x + u_0, & u_m \neq 0, \\ f_2 = v_n x^n + v_{n-1} x^{n-1} + \dots + v_1 x + v_0, & v_n \neq 0. \end{cases} \quad (8)$$

The Bezout matrix $\hat{B}(f_1, f_2) = (\hat{b}_{ij})$ is defined by

$$\hat{b}_{ij} = |u_0 v_{i+j-1}| + |u_1 v_{i+j-2}| + \dots + |u_k v_{i+j-k-1}|,$$

where $|u_r v_s| = u_s v_r - u_r v_s$, $k = \min(i-1, j-1)$ and $v_r = 0$

if $r > n$ [3]. It satisfies that

$$\frac{f_1(x)f_2(y) - f_1(y)f_2(x)}{x - y} = [1, x, x^2, \dots, x^{m-1}] \hat{B}(f_1, f_2) [1, y, y^2, \dots, y^{m-1}]^T.$$

Notice that the Bezout matrix $B(f_1, f_2)$ defined in Maple is as follows:

$$B(f_1, f_2) = -J\hat{B}(f_1, f_2)J, \quad (9)$$

where J is an anti-diagonal matrix with 1 as its nonzero entries.

Theorem 1. [3] *Given univariate polynomials $f_1, f_2 \in \mathbb{C}[x]$ with $\deg(f_1) = m$, $\deg(f_2) = n$, $m \geq n$, then we have*

$$\dim \text{NullSpace}(B(f_1, f_2)) = \deg(\gcd(f_1, f_2)).$$

Theorem 2. [37] *Given univariate polynomials $f_1(x), f_2(x)$ with $\deg(f_1) = m$, $\deg(f_2) = n$, $m \geq n$. Let $p(x) = \gcd(f_1, f_2)$ with $\deg(p) = r$, then we have*

1. $\text{rank}(B(f_1, f_2)) = m - r$, and $\det(B(f_1, f_2)_k) \neq 0$ for $k \leq m - r$, where $B(f_1, f_2)_k$ is the $k \times k$ leading principal submatrix of $B(f_1, f_2)$, but $\det(B(f_1, f_2)_k) = 0$ for $k > m - r$.
2. Suppose $\mathbf{y} = [y_0, y_1, \dots, y_{m-r-1}]^T$ satisfies $C\mathbf{y} = \mathbf{b}$, where $C = B(f_1, f_2)_{m-r}$ and $-\mathbf{b}$ is a vector formed from the first $m - r$ entries of the $m - r + 1$ -th column of $B(f_1, f_2)$. Let $\mathbf{u} = (u_i)_{i=0, \dots, m-r} = (JB(f_1, 1))_{m-r+1} \cdot [y_0, \dots, y_{m-r}]^T$, $y_{m-r} = 1$, then $f_1(x) = p(x)u(x)$, where $u(x) = \sum_{i=0}^{m-r} u_i x^i$.

According to Theorem 2, we can estimate efficiently the degree r of $\gcd(f_1, f_2)$ by checking whether the first $1 \times 1, 2 \times 2, 4 \times 4, \dots, 2^{\lceil \log_2(m-r+1) \rceil} \times 2^{\lceil \log_2(m-r+1) \rceil}$ leading principal submatrices are singular. Suppose $r = \deg(\gcd(f_1, f_2))$, we compute the cofactor $u(x)$ by solving an $(m - r) \times (m - r)$ linear system and the GCD can be found using the approximate polynomial division based on FFT. It is clear that we only need to form the first $(m - r + 1) \times (m - r + 1)$ submatrix of $B(f_1, f_2)$. This will save a lot of time and space when $m - r \ll m$. For example, since the blurring function has very low degree, the GCD problem arising from reconstructing an image from distorted images will satisfy $m - r \ll m$.

We recall how to compute the unknown factor $p(x)$ from polynomials $f(x)$ and $u(x)$ by the DFT algorithm. The associated convolution form of $f(x) = p(x)u(x)$ is

$$\mathbf{f} = \mathbf{p} * \mathbf{u},$$

where $\mathbf{f}, \mathbf{p}, \mathbf{u} \in \mathbb{C}^m$ are coefficient vectors of polynomials f, p, u respectively and $m = \deg(f) + 1$. Evaluating of $f(x)$ and $u(x)$ at $x_k = e^{-\frac{2k\pi i}{m}}$, $0 \leq k \leq m - 1$, we obtain the evaluation vector

$$[p(x_0), p(x_1), \dots, p(x_{m-1})]^T,$$

where $p(x_k) = f(x_k)/u(x_k)$, for $0 \leq k \leq m - 1$. The coefficient vector $\mathbf{p}$ associated with $p(x)$ is obtained by applying inverse DFT to $[p(x_0), p(x_1), \dots, p(x_{m-1})]^T$. The complexity of the polynomial division based on DFT for univariate polynomials is $O(m^2)$. The complexity of the fast algorithm using FFT for the univariate polynomial division is $O(m \log(m))$ [33, 34].

Now let us consider the general case, i.e., the condition $m - r \ll m$ does not hold. It is well known that the displacement rank of an $m \times m$ Bezout matrix is 2:

$$\nabla B = B - Z_1 B Z_0^T, \text{ and } \text{rank}(\nabla B) = 2, \quad (10)$$

where Z_a is the unit a -circulant matrix [34],

$$Z_a = \begin{bmatrix} 0 & & & a \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix}.$$

Hence, it is also possible to apply fast algorithms to compute the GCD of univariate polynomials. The complexity of the fast algorithm given in [5] is $O(m^2)$. A more extensive description about fast algorithms for matrices with low displacement rank can be found in [21, 34].

Although we have a fast Bezout-type GCD algorithm, it has been pointed out in Section 1, the cost to generate the $m \times m$ matrix $B(f_1, f_2)$ is $O(m^2)$, which will be expensive if m is over hundreds. On the contrary, the Sylvester matrix $S(f_1, f_2)$ is:

$$\begin{bmatrix} u_m & u_{m-1} & \cdots & u_1 & u_0 & & & \\ & u_m & u_{m-1} & \cdots & \ddots & \ddots & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & u_m & u_{m-1} & \cdots & u_1 & u_0 \\ v_n & v_{n-1} & \cdots & v_1 & v_0 & & & \\ & v_n & v_{n-1} & & \ddots & \ddots & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & v_n & v_{n-1} & \cdots & v_1 & v_0 \end{bmatrix}.$$

It costs almost nothing to generate the Sylvester matrix. Moreover, we know that the Sylvester matrix S is a quasi-Toeplitz matrix, i.e., the matrix $S - Z_0 S Z_0^T$ has rank at most 2.

Fast GCD algorithms with complexity $O(m^2)$ have been given in [5, 29, 30, 41, 43]. Hence, when the condition $m - r \ll m$ does not hold, it would be more efficient to construct the Sylvester matrix of f_1 and f_2 and use the fast Sylvester-type GCD algorithms.

Algorithm Approximate Univariate Polynomial GCD

Input: $\triangleright f_1(x), f_2(x) \in \mathbb{C}[x]$ with $m = \deg(f_1), n = \deg(f_2)$ and $m \geq n$.
 $\triangleright \epsilon \in \mathbb{R}_{>0}$: the given tolerance.
Output: $\triangleright p(x) \in \mathbb{C}[x]$: an approximate GCD of f_1 and f_2 .
 $\triangleright u(x), v(x) \in \mathbb{C}[x]$: approximate cofactors corresponding to f_1, f_2 .

Case 1: $m - r \ll m$ holds.

UH1 Estimate the degree r of $\gcd(f_1, f_2)$ for the given tolerance ϵ , and form C and $\mathbf{b}$ as shown in Theorem 2.

UH2 Obtain $u(x)$ by solving $C\mathbf{y} = \mathbf{b}$.

UH3 Compute $p(x)$ and $v(x)$ by applying the fast polynomial division based on FFT to $f_1(x)$ and $u(x)$; $f_2(x)$ and $p(x)$ respectively.

Case 2: $m - r \ll m$ does not hold.

UL1 Compute $u(x)$ via a fast GCD algorithm based on the displacement structure of the Sylvester matrix.

UL2 Compute $p(x)$ and $v(x)$ by applying the fast polynomial division based on FFT to $f_1(x)$ and $u(x)$; $f_2(x)$ and $p(x)$ respectively.

□

Theorem 3. *Given two univariate polynomials $f_1, f_2 \in \mathbb{C}[x]$ with $m = \max(\deg(f_1), \deg(f_2))$, $r = \deg(\gcd(f_1, f_2))$. In Case 1, if $m - r = O(m^{1/3})$, the total number of operations for computing the approximate GCD of univariate polynomials f_1 and f_2 based on the leading $(m - r + 1) \times (m - r + 1)$ principal submatrix of the Bezout matrix and FFT is $O(m \log(m))$. In Case 2, using the fast GCD algorithms based on the displacement structure of the Sylvester matrix, the total number of operations for computing the GCD is $O(m^2)$.*

Proof.

- Case 1. According to Theorem 2, we need to check whether the $k \times k$ leading principal submatrices are singular for $k \leq m - r + 1$. The number of operations involved in step UH1 is bounded by $O((m - r + 1)^3 \log(m - r + 1))$. It takes $O((m - r + 1)^3)$ operations to solve the linear system in step UH2. The fast polynomial divisions based on FFT in step UH3 cost $O(m \log(m))$. If $m - r = O(m^{1/3})$, then the operations in all three steps are bounded by $O(m \log(m))$.
- Case 2. The number of operations need in step UL1 is bounded by $O(m^2)$ since the Sylvester matrix has displacement rank 2.

□

3. AN APPROXIMATE BIVARIATE POLYNOMIAL GCD ALGORITHM

Suppose $f_1(x, y)$ and $f_2(x, y)$ are given in (4). We assume $\deg_x(f_1) = \deg_x(f_2) = m$ and $\deg_y(f_1) = \deg_y(f_2) = n$. In [35], they substituted

$$x = e^{-\frac{2k\pi i}{m}}, \quad k = 0, 1, \dots, m - 1$$

into f_1 and f_2 . For each k , this results in two univariate polynomials $f_1(e^{-\frac{2k\pi i}{m}}, y)$ and $f_2(e^{-\frac{2k\pi i}{m}}, y)$, applying the approximate GCD algorithm to these univariate polynomials, they obtained scaled quantity $c_0(e^{-\frac{2k\pi i}{m}})p(e^{-\frac{2k\pi i}{m}}, y)$. They proceeded further by substituting

$$y = e^{-\frac{2l\pi i}{n}}, \quad l = 0, 1, \dots, n - 1$$

and formed a matrix of discrete Fourier transform elements,

$$A(k, l)a(k) = p(e^{-\frac{2k\pi i}{m}}, e^{-\frac{2l\pi i}{n}}), \quad (11)$$

where $A(k, l)$ is the evaluation of the GCD of $f_1(e^{-\frac{2k\pi i}{m}}, y)$ and $f_2(e^{-\frac{2k\pi i}{m}}, y)$. Carrying out similar operations by substituting $y = e^{-\frac{2l\pi i}{n}}$ into f_1 and f_2 , taking their univariate GCD and further substituting $x = e^{-\frac{2k\pi i}{m}}$, they obtained another matrix

$$B(k, l)b(l) = p(e^{-\frac{2k\pi i}{m}}, e^{-\frac{2l\pi i}{n}}). \quad (12)$$

The vectors $a(k)$ and $b(l)$ are obtained by solving the least squares problem

$$A(k, l)a(k) - B(k, l)b(l) = 0, \quad (13)$$

and therefore, $p(x, y) = \gcd(f_1, f_2)$ can be computed by applying inverse DFT to

$$p(e^{-\frac{2k\pi i}{m}}, e^{-\frac{2l\pi i}{n}}) = \frac{1}{2}(A(k, l)a(k) + B(k, l)b(l)). \quad (14)$$

For images of size $n \times n$, their algorithm requires $O(n^4)$ operations.

For polynomials arising from blurred images, as we have mentioned in Section 1, the blurring function always has very low degree. Hence the degrees in variables x and y of the approximate GCD of $f_1(x, y)$ and $f_2(x, y)$ are almost as large as m and n . It would be much cheaper if we interpolate cofactor $u(x, y)$ or $v(x, y)$ by the above procedure, and then compute the GCD $p(x, y)$ by applying the fast polynomial division to $f_1(x, y)$ or $f_2(x, y)$. The division of two bivariate polynomials can also be done by the fast algorithm based on FFT. Hence, we can reduce the cost of the algorithm to $O(n^2 \log(n))$ for identifying both the true image and the blurring functions from blurred images of size $n \times n$ when the blurring functions have very low degree.

Algorithm Approximate Bivariate Polynomial GCD

Input: $\triangleright f_1(x, y), f_2(x, y) \in \mathbb{C}[x, y]$ with $\deg_x(f_1) = \deg_x(f_2) = m$ and $\deg_y(f_1) = \deg_y(f_2) = n$.
 $\triangleright \epsilon \in \mathbb{R}_{>0}$: the given tolerance.

Output: $\triangleright p(x, y) \in \mathbb{C}[x, y]$: an approximate GCD of f_1 and f_2 .
 $\triangleright u(x, y), v(x, y) \in \mathbb{C}[x, y]$: approximate cofactors of f_1 and f_2 respectively.

I. Estimate $r = \deg_x(p)$ and $s = \deg_y(p)$ for the given tolerance ϵ .

II. **Case 1:** $m - r + n - s \ll m + n$.

BH1 Apply Algorithm Approximate Univariate Polynomial GCD (Case 1) to compute the evaluation matrix $[u(x_k, y_l)] \in \mathbb{C}^{(m-r+1) \times (n-s+1)}$, where

$$x_k = e^{-\frac{2k\pi i}{m-r+1}}, \quad 0 \leq k \leq m - r, \\ y_l = e^{-\frac{2l\pi i}{n-s+1}}, \quad 0 \leq l \leq n - s.$$

BH2 Apply inverse FFT to $[u(x_k, y_l)]$ to compute $u(x, y)$.

BH3 Compute $p(x, y)$ by applying the fast polynomial division to $f_1(x, y)$ and $u(x, y)$.

BH4 Compute $v(x, y)$ by applying the fast polynomial division to $f_2(x, y)$ and $p(x, y)$.

Case 2: $r + s \ll m + n$.

BL1 Apply Algorithm Approximate Univariate Polynomial GCD (Case 2) to compute the evaluation matrix $[p(x_k, y_l)] \in \mathbb{C}^{(r+1) \times (s+1)}$, where

$$x_k = e^{-\frac{2k\pi i}{r+1}}, \quad 0 \leq k \leq r, \\ y_l = e^{-\frac{2l\pi i}{s+1}}, \quad 0 \leq l \leq s.$$

BL2 Apply inverse FFT to the matrix $[p(x_k, y_l)]$ to get $p(x, y) = \gcd(f_1, f_2)$.

BL3 Compute $u(x, y), v(x, y)$ by applying the fast polynomial division to polynomials $f_1(x, y), p(x, y)$ and $f_2(x, y), p(x, y)$ respectively.

Theorem 4. *Given bivariate polynomials $f_1, f_2 \in \mathbb{C}[x, y]$, we assume that $n = \deg_x(f_1) = \deg_x(f_2) = \deg_y(f_1) = \deg_y(f_2)$, $r = \deg_x(\gcd(f_1, f_2)) = \deg_y(\gcd(f_1, f_2))$. The total number of operations for computing the approximate GCD of polynomials f_1 and f_2 is bounded by $O(n^2 \log(n))$ when $n-r = O(n^{1/2})$ or $r = O(\log(n))$ which corresponding to Case 1 and Case 2 respectively.*

Proof.

- According to Theorem 3, the number of operations for estimating r and s in step I is bounded by $O(n^2)$.
- Case 1. It takes $O(n^2 \log(n))$ operations to get the evaluations of polynomials f_1 and f_2 at the DFT points. We need $O((n-r)^4)$ operations to get the evaluation matrix $[u(x_k, y_l)]$ by step UH2 of Algorithm Approximate Univariate Polynomial GCD (Case 1). If $n-r = O(n^{1/2})$, then the number of operations involved in step BH1 is bounded by $O(n^2 \log(n) + n^2)$. Moreover, if we apply the fast polynomial division based on FFT in steps BH2, BH3 and BH4, the total number of operations involved in these three steps is bounded by $O(n^2 \log(n))$.
- Case 2. We compute the evaluation matrix $[p(x_k, y_l)]$ via the fast GCD algorithm based on the displacement structure of the Sylvester matrix. Hence, the complexity of BL1 is bounded by $O(m^2 + n^2 \log(n))$. If $r = O(\log(n))$, then the complexity of BL1 is bounded by $O(n^2 \log(n))$. The total number of operations of steps BL2 and BL3 is bounded by $O(n^2 \log(n))$ if we apply the fast polynomial division based on FFT.

□

4. EXPERIMENTS

The following examples come from literatures on image deconvolution. We show that our new algorithm implemented in Maple can reconstruct true images from blurred images successfully in few seconds. The algorithm in [35] has also been implemented in Maple. All experiments are run on an Inter(R) Core(TM) 2 Quad Cpu at 2.40 GHz for Digits=14 in Maple 13 under Windows.

Example 1. *(Reconstructing Image From Three Distorted Images)*

In Figure 1, Figure 1.(a) is an image of size 250×250 scanned from [7]. Figure 1.(b) contains three distorted images built by convolving Figure 1.(a) with three 7×7 co-prime distortion filters. Since we can not add white noise in Maple directly, for simplicity, we add random noise with $\text{SNR} = 52\text{dB}$.

Figure 1.(c) is the image reconstructed by running Algorithm Approximate Bivariate Polynomial GCD (Case 1, $7+7 \ll 250+250$) in about 1.10 seconds; whereas the time obtained by running the algorithm [35] in Maple is 46.90 seconds.

Example 2. *(Reconstructing RGB Image From Two Distorted Images)*

Size	Three Grey		Two RGB	
	Time(s) (Bezout)	Time(s) (Liang)	Time(s) (Bezout)	Time(s) (Liang)
256×256	1.23	49.34	1.66	152.40
512×512	5.72	317.15	7.66	982.96
1024×1024	28.63	2995.16	45.42	7914.44

Table 1: Algorithm performance on benchmark

In Figure 2, Figure 2.(a) is an image of size 128×170 scanned from [35]. Figure 2.(b) and Figure 2.(c) are two distorted images built by convolving Figure 2.(a) with two 7×7 co-prime distortion filters and with the additive noise $\text{SNR} = 52\text{dB}$.

Figure 2.(d) is the image reconstructed successfully by running Algorithm Approximate Bivariate Polynomial GCD (Case 1, $7+7 \ll 128+170$) in about 0.73 seconds; whereas the time obtained by running the algorithm in [35] is 47.26 seconds.

In Table 1, we show the performance of our algorithm for recovering large images obtained by convolving three original images downloaded from <http://sipi.usc.edu/database/> with distortion filters of sizes $7 \times 7, 13 \times 13, 19 \times 19$ respectively. The additive noises are the same: $\text{SNR} = 63\text{dB}$. Here *Size* denote the size of the original images; *Time(Bezout)* and *Time(Liang)* contain the timings to solve blind image deconvolution problem by running our new algorithm based on Bezout-type GCD algorithm and using the algorithm in [35], respectively.

Remark 1. In [35], they also considered the case where there was only one available grey image. They assumed the blurring function is one dimension. By z -transforms, the polynomial form of (1) can be written as :

$$f(x, y) = p(x, y)u(y) + n(x, y), \quad (15)$$

where f, p, u, n are z -transforms of F, P, U, N respectively. Suppose $f(x, y)$ can be represented as follows:

$$f(x, y) = \sum_{i=0}^m a_i(y)x^i,$$

where $m = \deg_x(f)$. Assuming $p(x, y)$ is primitive with respect to x , then $u(y)$ is the common divisor of $a_0(y), \dots, a_m(y)$. The GCD polynomial $p(x, y)$ can be obtained by performing the polynomial division of $f(x, y)$ and $u(y)$. In the approximate case, we need to compute approximate univariate GCDs of several polynomials (more than 2), which can be solved by the following two options:

1. Choose random values $s_0, s_1, \dots, s_m, t_0, t_1, \dots, t_m$ and get two polynomials

$$f_1 = \sum_{i=0}^m s_i a_i(y), \quad \text{and} \quad f_2 = \sum_{i=0}^m t_i a_i(y).$$

Compute the approximate GCD $u'(y)$ of f_1 and f_2 , with high probability, we have $u'(y) = \gcd(a_0, \dots, a_m)$.

2. Construct the generalized Bezout matrix $B(a_0, \dots, a_m)$ or the generalized Sylvester matrix $S(a_0, \dots, a_m)$ to compute the approximate GCD.

□

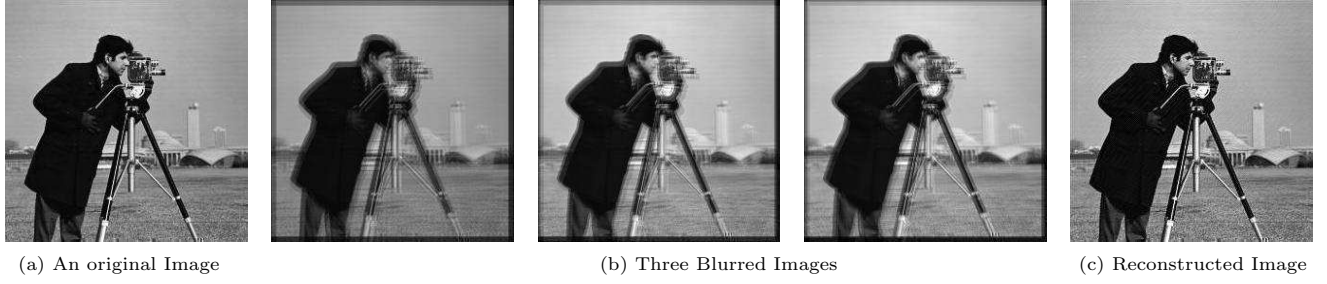


Figure 1: Blind deblurring from three distorted and noisy ($\text{SNR} = 52\text{dB}$) images



Figure 2: Blind deblurring from two distorted and noisy ($\text{SNR} = 52\text{dB}$) RGB images

Example 3. (*One Blurred Grey Image*)

In Figure 3, Figure 3.(a) is an image of size 337×77 scanned from [35]. Figure 3.(b) is an image of size 312×77 reconstructed in about 3.17 seconds by using the Sylvester-type GCD algorithm for univariate polynomials and option 1. Notice that for this example, we need to generate a 336×336 Bezout matrix. It takes already more than 5 seconds in Maple to generate the matrix. Whereas the time obtained by running the algorithm in [35] is 3.46 seconds.

Example 4. (*Reconstructing RGB Image from One Blurred Image*)

Figure 4.(a) is an image of size 128×170 which is scanned from [35]. The distorted image of Figure 4.(b) is built by convolving Figure 4.(a) with a 7×7 distortion filter and with the additive noise $\text{SNR} = 53\text{dB}$. Figure 4.(c) is the image reconstructed by running Algorithm Approximate Bivariate Polynomial GCD (Case 2, $7 + 7 \ll 128 + 170$) in about 5.85 seconds if we are using the Sylvester-type GCD algorithm for univariate polynomials. Whereas the time obtained by running the algorithm in [35] is 118.47 seconds. $\square$

5. CONCLUSION

In this paper, we present a specialized algorithm for computing approximate GCDs of univariate or bivariate polynomials arising from blind images deconvolution problem. To recover images from the blurred ones of size $n \times n$, we are able to reduce the complexity to $O(n^2 \log(n))$ in the case of blurring functions of very low degree.

We have implemented both Bezout-type and Sylvester-type univariate GCD algorithms, the fast polynomial division and interpolation based on FFT in Maple. We show that our algorithm is efficient and quite robust. When the additive noise satisfies $\text{SNR} \geq 50\text{dB}$, i.e., the relative er-

rors of polynomials corresponding to the distorted images are within 10^{-4} , our new algorithm can recover successfully true images from blurred and distorted images. However, we also notice that when the additive noise is smaller than 50dB , it is still hard to use our algorithm to recover original images from distorted ones. Moreover, although our new algorithm is much faster than the one in [23] for computing GCDs of bivariate polynomials, unfortunately, the backward error becomes much bigger. We may have to sacrifice the efficiency to have a more robust GCD algorithm. Some new techniques in computing GCDs [5, 23, 25, 30, 39] of polynomials and multivariate polynomial interpolation [18, 22, 24] may be used to improve the efficiency and stability of our algorithm.

Acknowledgments

We thank John May for valuable suggestions that improve our Maple code significantly. The authors also wish to thank the reviewers for their helpful comments.

6. REFERENCES

- [1] BARNETT, S. Greatest common divisor of two polynomials. *Linear Algebra Appl.* 3 (1970), 7–9.
- [2] BARNETT, S. Greatest common divisor of several polynomials. *Proc. Camb. Phil. Soc.* 70 (1971), 263–268.
- [3] BARNETT, S. A note on the Bezoutian matrix. *SIAM J. APPL. MATH.* 22, 1 (January 1972), 84–86.
- [4] BECKERMANN, B., AND LABAHN, G. When are two polynomials relatively prime? *J. Symbolic Comput.* 26 (1998), 677–689.
- [5] BINI, D. A., AND BOITO, P. Structured matrix-based methods for polynomial ϵ -gcd: analysis and comparisons. In Brown [6].

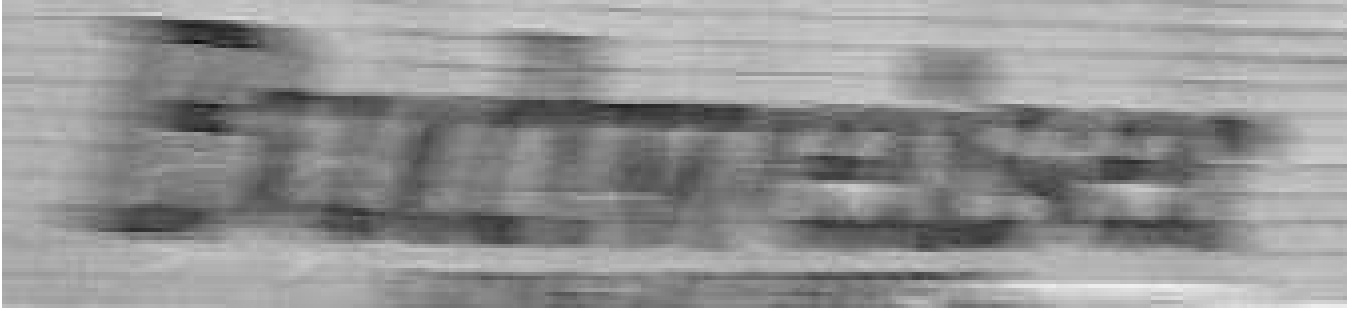


Figure (a): Blurred Image



Figure (b): Reconstructed Image

Figure 3: Blind deblurring of linear motion blurred image



Figure 4: Blind deblurring from one distorted and noisy (SNR = 53dB) RGB images

- [6] BROWN, C. W., Ed. *ISSAC 2007 Proc. 2007 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2007), ACM Press.
- [7] CAMPISI, P., AND EGIAZARIAN, K. *Blind image deconvolution: theory and applications*. CRC Press, 2007.
- [8] CORLESS, R., GIANNI, P., TRAGER, B., AND WATT, S. The singular value decomposition for polynomial systems. In *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. ISSAC'95* (New York, 1995), A. Levelt, Ed., ACM Press, pp. 96–103.
- [9] CORLESS, R., WATT, S., AND ZHI, L. QR factoring to compute the GCD of univariate approximate polynomials. *IEEE Transactions on Signal Processing* 52 (Dec 2004), 3394–3402.
- [10] DIAZ-TOCA, G., AND GONZALEZ-VEGA, L. Computing greatest common divisors and squarefree decompositions through matrix methods: The parametric and approximate cases. *Linear Algebra Appl.* 412 (2006), 222–246.
- [11] DIAZ-TOCA, G. M., AND GONZALEZ-VEGA, L. Barnett's theorems about the greatest common divisor of several univariate polynomials through Bezout-like matrices. *Journal of Symbolic Computation* 34, 1 (2002), 59 – 81.
- [12] DUMAS, J.-G., Ed. *ISSAC MMVI Proc. 2006 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2006), ACM Press.
- [13] EMIRIS, I., GALLIGO, A., AND LOMBARDI, H. Certified approximate univariate GCDs. *J. Pure Applied Algebra*. 117 & 118 (May 1996), 229–251. Special Issue on Algorithms for Algebra.
- [14] GAO, S., KALTOFEN, E., MAY, J., YANG, Z., AND ZHI, L. Approximate factorization of multivariate polynomials via differential equations. In Gutierrez [19], pp. 167–174.
- [15] GEMIGNANI, L. Gcd of polynomials and Bezout matrices. In *Proc. 1997 Internat. Symp. Symbolic*

- Algebraic Comput. ISSAC'97* (New York, 1997), Küchlin, Ed., ACM Press, pp. 271–277.
- [16] GHIGLIA, D. C., ROMERO, L. A., AND MASTIN, G. A. Systematic approach to two-dimensional blind deconvolution by zero-sheet separation. *J. Opt. Soc. Am. A* 10 (1993), 1024–1036.
 - [17] GIANNAKIS, G., AND HEATH, R. Blind identification of multichannel fir blurs and perfect image restoration. *IEEE Trans. Image Processing* 9 (2000), 1877–1896.
 - [18] GIESBRECHT, M., LABAHN, G., AND LEE, W. Symbolic-numeric sparse interpolation of multivariate polynomials. In Dumas [12], pp. 116–123.
 - [19] GUTIERREZ, J., Ed. *ISSAC 2004 Proc. 2004 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2004), ACM Press.
 - [20] HRIBERNIG, V., AND STETTER, H. Detection and validation of clusters of polynomials zeros. *J. Symbolic Comput.* 24 (1997), 667–681.
 - [21] KAILATH, T., AND SAYED, A. H. Displacement structure: theory and applications. *SIAM Review* 37, 3 (1995), 297–386.
 - [22] KALTOFEN, E., AND YANG, Z. On exact and approximate interpolation of sparse rational functions. In Brown [6], pp. 203–210.
 - [23] KALTOFEN, E., YANG, Z., AND ZHI, L. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In Dumas [12], pp. 169–176. Full version, 21 pages. Submitted, December 2006.
 - [24] KALTOFEN, E., YANG, Z., AND ZHI, L. On probabilistic analysis of randomization in hybrid symbolic-numeric algorithms. In *SNC'07 Proc. 2007 Internat. Workshop on Symbolic-Numeric Comput.* (New York, N. Y., 2007), J. Verschelde and S. M. Watt, Eds., ACM Press, pp. 11–17.
 - [25] KALTOFEN, E., YANG, Z., AND ZHI, L. Structured low rank approximation of a Sylvester matrix. In *Symbolic-Numeric Computation*, D. Wang and L. Zhi, Eds., Trends in Mathematics. Birkhäuser Verlag, Basel, Switzerland, 2007, pp. 69–83. Preliminary version in [40], pp. 188–201.
 - [26] KARMAKAR, N., AND LAKSHMAN Y. N. Approximate polynomial greatest common divisors and nearest singular polynomials. In *ISSAC 96 Proc. 1996 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 1996), Lakshman Y. N., Ed., ACM Press, pp. 35–42.
 - [27] KARMAKAR, N. K., AND LAKSHMAN Y. N. On approximate GCDs of univariate polynomials. *J. Symbolic Comput.* 26, 6 (1998), 653–666. Special issue on Symbolic Numeric Algebra for Polynomials S. M. Watt and H. J. Stetter, editors.
 - [28] LANE, R. G., AND BATES, R. H. T. Automatic multidimensional deconvolution. *J. Opt. Soc. Am. A* 4 (1987), 180–188.
 - [29] LI, B., LIU, Z., AND ZHI, L. A structured rank-revealing method for sylvester matrix. *J. Comput. Appl. Math.* 213, 1 (2008), 212–223.
 - [30] LI, B., YANG, Z., AND ZHI, L. Fast low rank approximation of a Sylvester matrix by structured total least norm. *J. ISSAC (Japan Society for Symbolic and Algebraic Computation)* 11, 3,4 (2005), 165–174.
 - [31] MARKOVSKY, I., AND HUFFEL, S. V. An algorithm for approximate common divisor computation. Internal Report 05-248, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2005.
 - [32] NODA, M., AND SASAKI, T. Approximate GCD and its application to ill-conditioned algebraic equations. *J. Comput. Appl. Math.* 38 (1991), 335–351.
 - [33] P. DUHAMEL, M. V. Fast Fourier Transforms: A tutorial review and a state of the art. *Signal Processing* 19 (April 1990), 259–299.
 - [34] PAN, V. Numerical computation of a polynomial GCD and extensions. *Information and computation* 167 (2001), 71–85.
 - [35] PILLAI, S. U., AND LIANG, B. Blind image deconvolution using a robust GCD approach. *IEEE Transactions on Image Processing* 8, 2 (1999), 295–301.
 - [36] SCHÖNHAGE, A. Quasi-gcd computations. *Journal of Complexity* 1 (1985), 118–137.
 - [37] SUN, D., AND ZHI, L. Structured low rank approximation of a Bezout matrix. *MM Research Preprints* 25 (December 2006), 207–218.
 - [38] SUN, D., AND ZHI, L. Structured low rank approximation of a Bezout matrix. *Mathematics in Computer Science* 1, 2 (2007), 427–437.
 - [39] TERUI, A. An iterative method for calculating approximate GCD of univariate polynomials. In *ISSAC 2009 Proc. 2009 Internat. Symp. Symbolic Algebraic Comput.* (New York, NY, USA, 2009), J. May, Ed., ACM, pp. 351–358.
 - [40] WANG, D., AND ZHI, L., Eds. *Internat. Workshop on Symbolic-Numeric Comput. SNC 2005 Proc.* (2005). Distributed at the Workshop in Xi'an, China, July 19–21.
 - [41] ZAROWSKI, C. J., MA, X., AND FAIRMAN, F. W. A QR-factorization method for computing the greatest common divisor of polynomials with real-valued coefficients. *IEEE Trans. Signal Processing* 48 (2000), 3042–3051.
 - [42] ZENG, Z., AND DAYTON, B. The approximate GCD of inexact polynomials part II: a multivariate algorithm. In Gutierrez [19], pp. 320–327.
 - [43] ZHI, L. Displacement structure in computing approximate GCD of univariate polynomials. In *Proc. Sixth Asian Symposium on Computer Mathematics (ASCM 2003)* (Singapore, 2003), Z. Li and W. Sit, Eds., vol. 10 of *Lecture Notes Series on Computing*, World Scientific, pp. 288–298.

Polynomial Integration on Regions Defined by a Triangle and a Conic

David Sevilla and Daniel Wachsmuth
Johann Radon Institute for Computational and Applied Mathematics (RICAM)
Austrian Academy of Sciences
Altenbergerstrasse 69
A-4040 Linz, Austria
david.sevilla@oeaw.ac.at , daniel.wachsmuth@oeaw.ac.at

ABSTRACT

We present an efficient solution to the following problem, of relevance in a numerical optimization scheme: calculation of integrals of the type

$$\iint_{T \cap \{f \geq 0\}} \phi_1 \phi_2 \, dx \, dy$$

for quadratic polynomials f, ϕ_1, ϕ_2 on a plane triangle T . The naive approach would involve consideration of the many possible shapes of $T \cap \{f \geq 0\}$ (possibly after a convenient transformation) and parameterizing its border, in order to integrate the variables separately. Our solution involves partitioning the triangle into smaller triangles on which integration is much simpler.

Categories and Subject Descriptors

G.1.8 [Numerical Analysis]: Partial Differential Equations—*finite element methods*; I.1.2 [Symbolic and algebraic manipulation]: Algorithms—*algebraic algorithms*

General Terms

Theory

Keywords

symbolic integration, triangular subdivision, optimal control, variational discretization, quadratic shape functions

1. INTRODUCTION

This article presents a symbolic solution to a problem of relevance in a numerical optimization scheme: the numerical solution of optimal control problems with partial differential equations as constraint requires to discretize the problem, i.e. to solve finite-dimensional approximations, see e.g. [4]. When applying the variational discretization concept [3], the

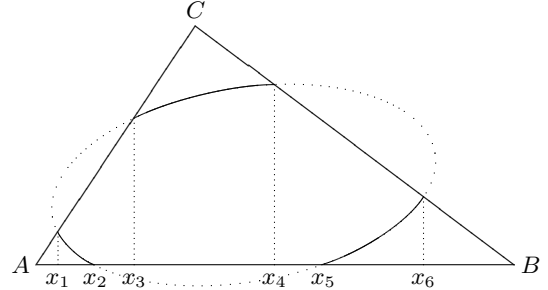
following problem arises: integrals of the type

$$\iint_{T \cap \{f \geq 0\}} g(x, y) \, dx \, dy, \quad g = \phi_1 \cdot \phi_2 \quad (1)$$

for quadratic polynomials f, ϕ_1, ϕ_2 on a plane triangle T have to be evaluated accurately. Up to now the variational discretization method was used only for degree 1, i.e. where the function f defining the integral region in (1) is a polynomial of degree 1. Using polynomials with higher order gives better approximation results, see Theorem 5.5 below.

The naive approach to compute the integral (1) would involve consideration of the many possible shapes of $T \cap \{f \geq 0\}$ and parameterizing its border, in order to integrate the variables separately. This suffers from some computational difficulties as we show below.

EXAMPLE 1.1. Suppose that one side of the triangle lies on a horizontal line. Consider the situation where the region of integration is the part of the interior of an ellipse in the triangle, as in the figure.



We see that we could calculate the integral as the sum of five integrals on domains perpendicular to the x axis. For example the first one is

$$\int_{x_1}^{x_2} \left(\int_{c_-(x)}^{l_{AC}(x)} g(x, y) \, dy \right) dx$$

where $y = l_{AC}(x)$ is the equation of the line AC and $y = c_-(x)$ is the equation of the lower part of the ellipse. Note that we need to parameterize the ellipse (this will involve at best a square root or trigonometric functions) and calculate the x -coordinates of the relevant points, which also involve square roots. The value of the inner integral will be given by a formula of which an antiderivative must be computed then. The resulting formula is far from simple.

An alternative would be to apply an affine transformation so that the ellipse becomes a circle centered at the origin,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

followed by a change to polar coordinates. This does not make the integral significantly easier to compute.

And of course, here we use our knowledge of the relative position of the ellipse and the triangle, as in the figure; the possible relative positions of a conic and a triangle are many, and to discern them is not trivial.

In contrast, in the following particular case we obtain a simple formula.

EXAMPLE 1.2. Let $A = (0, 0)$, $B = (1, 0)$, $C = (1, 1)$, and assume that $f \geq 0$ on the triangle $T = ABC$. If $g(x, y) = \sum_{i+j \leq 4} b_{ij} x^i y^j$ then the integral becomes

$$\begin{aligned} \iint_T g(x, y) dx dy &= \int_0^1 \left(\int_0^x g(x, y) dy \right) dx = \\ &= \sum_{i+j \leq 4} \frac{b_{ij}}{(j+1)(i+j+2)} \quad (2) \end{aligned}$$

For a general triangle T , one applies an affine transformation which brings the vertices to the above points. The resulting integrand is a polynomial of the same degree, and only a constant factor is introduced by the substitution formula.

Our solution involves partitioning T into smaller triangles on which integration is much simpler. The result is a decision tree and several relatively simple explicit formulas, which form Algorithm 4.1. The particular nature of g , beyond it being a polynomial, will be immaterial. Our method could in principle be adapted for larger values of $\deg f$, although it may become too complicated for practical uses even for degree 3. Besides, only the quadratic case is relevant for the context in which this problem arose. It is important to point out that an implementation for floating point arithmetic would need a more detailed treatment, see the end of Section 5.

We describe the subdivision method in Section 2, the integration in the base cases in Section 3, the complete algorithm in Section 4, and a description of the application to optimization in Section 5 which includes some comments on the practical implementation of our algorithm.

2. TRIANGULAR SUBDIVISION

Our idea is to reduce the number of intersections between the curve $f = 0$ and the sides of the triangle, by cutting the triangle into pieces until we reach some base cases that we establish below. For those cases the integration will be much simpler than in Example 1.1. We leave for later the case where the conic $f = 0$ is degenerate (two lines, either intersecting, parallel, or coincident; one point; the empty set). Note that the type of a conic can be determined quickly by inspection of the equation.

First we introduce some nomenclature.

DEFINITION 2.1. Fix a nonsingular conic. A segment is called free (with respect to the conic) if it does not intersect it except possibly at the vertices of the segment.

REMARK 2.2.

1. Any line or segment intersects any conic at most at two points.

2. A segment joining two points of the conic is always free.

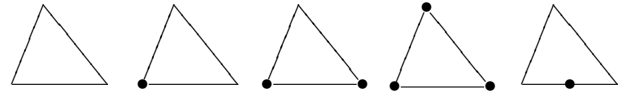
PROOF. Part 1 is a simple case of the weak Bézout's theorem [5]. Part 2 is a clear consequence of part 1. $\square$

The calculation of the intersections of a segment with a given conic (and thus the determination of the freedom of the segment) is straightforward, and fast in practice.

The next definition encapsulates the base cases of our subdivision method.

DEFINITION 2.3. A triangle is called free if either all its sides are free, or the intersection of its border with the conic is just one non-vertex point.

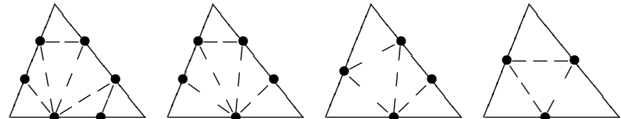
Thus there are five types of free triangles: those with all sides free and 0, 1, 2 or 3 intersections at the vertices; and those with no vertex intersections and one side intersection.



The rest of the section describes how to divide a given triangle so that all the pieces are free triangles. We proceed step by step in terms of the number of free sides.

LEMMA 2.4. Every triangle with no free sides can be cut into seven free triangles.

PROOF. Each non-free side has one or two interior intersections with the conic. We draw the four cases and one solution for each (possible vertex intersections are irrelevant here, thus not drawn). All the small triangles can be proven free by noting that their sides are either free parts of the original sides, or segments connecting two intersections (thus free by part 2 of Remark 2.2).

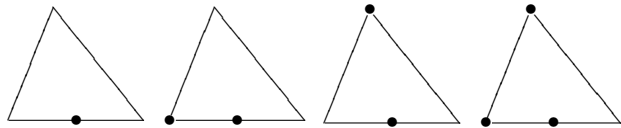


$\square$

The next step is to consider non-free triangles with one free side. We introduce another useful term.

DEFINITION 2.5. A triangle is almost free if exactly one of its sides is not free, and that side has only one intersection in its interior.

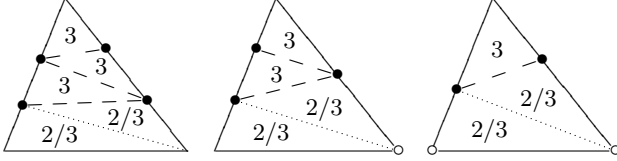
REMARK 2.6. There are four types of almost-free triangles, depending on the vertex intersections.



Note that if a triangle is almost free and has no vertex intersections with the conic, then it is free (first case).

LEMMA 2.7. Every triangle with exactly one free side can be cut into five free or almost-free triangles, with zero or two of them being almost free.

PROOF. There are three cases depending on the number of interior intersections with the non-free sides: $2 + 2$, $2 + 1$ and $1 + 1$. The diagrams show how to cut the triangle in the three cases (possible vertex intersections, marked in white, make no difference). The numbers of free sides in each piece are indicated. As before, if a segment intersects a conic in its endpoints then it is free by part 2 of Remark 2.2.



In all cases, the only segment that may not be free is the lowest new one (dotted), and it can only have one interior intersection (by part 1 of Remark 2.2, since one of its endpoints is in the conic already). $\square$

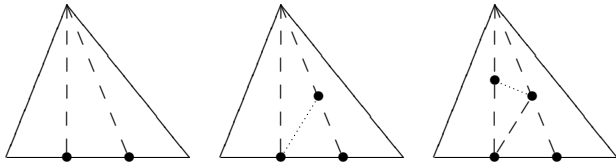
Next, we consider the case when two sides are free.

LEMMA 2.8. *Every triangle with exactly two free sides can be cut into four free or almost-free triangles. At least one of them is free, except if the original triangle is almost free.*

PROOF. The non-free side has one or two interior intersections; in the former case, the triangle is almost free and we are finished. If it has two interior intersections, we join them with the opposite vertex and create two interior sides. There are three possibilities:

1. If there are no interior intersections in the new sides, the three pieces are free.
2. If one of the new sides has one interior point, we obtain one free triangle and two almost-free triangles.
3. If both new sides have one interior point each, with one extra cut we obtain one free triangle and three almost-free triangles.

The dashed lines indicate the partitions described above.



Note that in the last two cases, cutting along the dotted line reduces the number of almost-free triangles by one, but it increases the total number of triangles. This might make a small difference in performance. $\square$

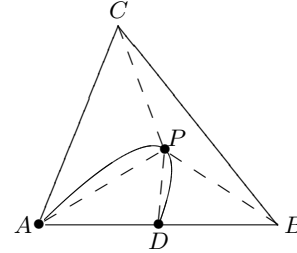
LEMMA 2.9. *Every almost-free triangle can be cut into four free triangles.*

PROOF. If no vertex is in the conic, the triangle is already free (first case of Remark 2.6). If the vertex opposite to the non-free side belongs to the conic (third and fourth cases of Remark 2.6) then the segment between them is free, and the triangle is cut into two free pieces.

There remains only one case (see figure below): the conic intersects the triangle at two points, a vertex A and an interior point D of the side AB . The conic cannot intersect AB tangentially (otherwise it would have multiplicity intersection ≥ 3 with that line). Therefore it must enter the triangle through D and it can only exit through A .

When CD is free, this segment cuts the triangle in two free pieces. However this is not true in general.

Choose any point P in the conic and inside the triangle, with the property that BP and CP are free; then the original triangle is cut in four free pieces.



It suffices that the tangent to the conic at P leaves B and C on the same half-plane: the branch of the conic must then be contained in the other half-plane, thus BP and CP will be free. We offer three such points which are efficiently computable: the point whose tangent is parallel to BC ; and the points at which the tangents pass through B or C . $\square$

Combining all the previous lemmas and counting the number of pieces at each step, we obtain the following result.

PROPOSITION 2.10. *Every triangle can be cut into eleven free triangles.*

REMARK 2.11. *It is possible to reduce the final number of free pieces to nine but one needs to use more often the re-course of finding tangency points in the conic as in Lemma 2.9; we chose the simpler approach. On the other hand, those points can be computed efficiently, which may make it attractive to minimize the number of triangles in practice. Still, the integration time in each piece depends on the particular intersections.*

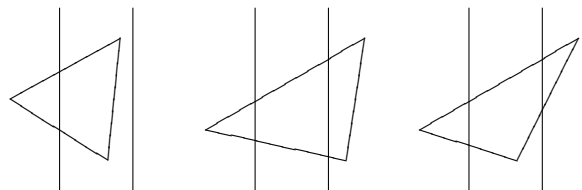
2.1 Degenerate conics

We analyze now how to calculate the integral when $f = 0$ is a degenerate conic. If it is empty, one point, or a double line, the integral is zero or the value on the full triangle.

2.1.1 Two parallel lines

If $f = 0$ is two parallel lines, it can be converted by an affine transformation into $x(x - 1) = 0$. We can determine in which of the regions $x \leq 0$, $0 \leq x \leq 1$, $1 \leq x$ lies the image of each vertex by looking at their x -coordinates.

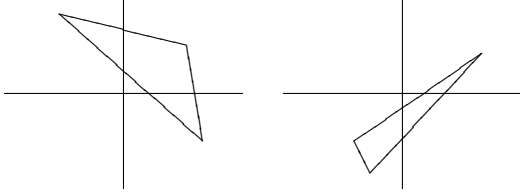
1. If all three vertices are in one of the three regions, the integral is either the full triangle integral or zero; we can determine the sign of f in the triangle and use (2).
2. Otherwise, the triangle is split into two or three pieces (not necessarily triangular). The figure below depicts the possible cases. Once we have determined on which region(s) we must integrate (the middle strip or its complement), this can be done solely by adding and subtracting integrals of triangular pieces, which can be calculated using (2).



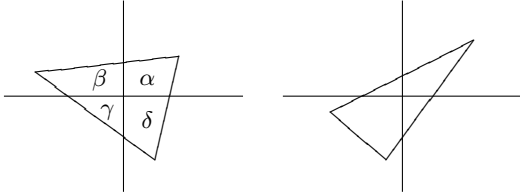
2.1.2 Two crossing lines

The conic can be transformed to the pair of lines $xy = 0$, what allows us to quickly determine in which quadrants the vertices lie. The region on which to integrate is the intersection of the triangle and two opposing quadrants.

1. If all vertices are in the same quadrant, the integral is the full triangle or zero.
2. If all vertices are in two adjacent quadrants, the triangle is divided in two pieces, one of which is a triangle (or both, if a vertex lies in the limiting line). The integral is that on the triangular piece, or the complementary.
3. If the triangle is divided in three pieces by the conic, either all vertices are in different regions, or they are in two opposing regions. In any case, we can compute the integral on the relevant region by adding and subtracting integrals on triangles.



4. Finally, if the triangle is divided in four pieces by the conic, there are two possible arrangements as well. Again, we can compute the integral by adding and subtracting triangles.



For example, in the left figure, the union of the top-right and bottom-left regions of the triangle is $\alpha + \gamma = (\alpha + \beta + \gamma + \delta) - (\beta + \gamma) - (\delta + \gamma) + 2\gamma$, where the terms in parenthesis, as well as γ , are triangles.

We can decide if we are in situation 1 or 2 by inspecting the signs of the coordinates of the transformed vertices. In order to differentiate situations 3 and 4 we use that the triangle is divided in four pieces if and only if the intersection of the two lines lies in its interior. This can be detected by calculating its barycentric coordinates as in the following algorithm.

ALGORITHM 2.12. Determine if a point P is inside a triangle ABC .

1. In the expression $\overrightarrow{AP} = \alpha \overrightarrow{AB} + \beta \overrightarrow{AC}$ calculate α and β :

$$\alpha = \frac{\det(\overrightarrow{AP}, \overrightarrow{AC})}{\det(\overrightarrow{AB}, \overrightarrow{AC})}, \quad \beta = \frac{\det(\overrightarrow{AB}, \overrightarrow{AP})}{\det(\overrightarrow{AB}, \overrightarrow{AC})}$$

where $\det(u, v) = u_1 v_2 - u_2 v_1$.

2. If $\alpha, \beta > 0$ and $\alpha + \beta < 1$ then P is contained in the triangle.

3. If $\alpha = 0$ and $\beta \in [0, 1]$; or $\beta = 0$ and $\alpha \in [0, 1]$; or $\alpha + \beta = 1$ and $\alpha \in [0, 1]$, then P is in the border of the triangle.

4. Otherwise P is outside the triangle.

In any case, for the final integral it is enough to add and subtract several instances of (2), with no subdivisions other than the given by the lines of the conic.

3. BASE CASE INTEGRATION

In this section we describe how to detect the relative position of the nondegenerate conic $f = 0$ and a free triangle ABC , and compute the integral, in the five possible cases of free triangles.

3.1 No intersections

There are three possibilities:

1. $T \subset \{f \geq 0\}$: the integral was computed in Example 1.2.
2. $T \subset \{f \leq 0\}$: the integral is zero.
3. $f = 0$ is an ellipse contained in T .

By inspecting f we can decide immediately whether $f = 0$ is not an ellipse, from which we would deduce that we are in the first or second case. Then one can discern by evaluating the sign of f at some interior point of the triangle.

On the other hand, if $f = 0$ is an ellipse, we have to determine if any of the two shapes is contained in the other. We can do this by mapping the ellipse to the unit circle.

ALGORITHM 3.1. Determine the relative position of the ellipse $f = 0$ and the triangle ABC , and the correct domain of integration.

1. Calculate an affine transformation $\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that sends $f = 0$ into $x^2 + y^2 = 1$. Let $P = (0, 0)$.
2. If $d(\phi(A), P) < 1$ then ABC is contained in the ellipse; evaluate the sign of f at some interior point of ABC to decide if the integral is the full triangle or zero.
3. Otherwise, decide if P is inside the triangle $A'B'C' := \phi(ABC)$ with Algorithm 2.12.
4. If P is in the triangle, then the ellipse is contained in it; evaluate the sign of f at $\phi^{-1}(P)$ to decide on which region to integrate.
5. Otherwise, none of the shapes contains the other; evaluate the sign of f at $\phi^{-1}(P)$ to decide if the integral is the full triangle or zero.

The remaining computation is the integral of g when the ellipse $f = 0$ is contained in the triangle. We show how to obtain a closed formula when $\{f \geq 0\}$ is the bounded region inside the ellipse; in the other case, the required integral is the difference of the full triangle integral and the former.

Let $\varphi = \phi^{-1}: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ which sends the circle $x^2 + y^2 = 1$ to f . Then

$$\iint_{\{f \geq 0\}} g \, dx \, dy = \iint_D g(\varphi) |J(\varphi)| \, dx \, dy$$

where D is the unit disc. Since φ is affine, $|J(\varphi)| \in \mathbb{R}$ and $\bar{g} := g(\varphi)$ is again a polynomial. Now, using polar coordinates, this is equal to

$$|J(\varphi)| \int_0^{2\pi} \left(\int_0^1 \bar{g}(r \cos \theta, r \sin \theta) \cdot r \, dr \right) d\theta$$

which is reduced to a linear combination of integrals of type $\int_0^{2\pi} \cos^i \theta \sin^j \theta \, d\theta$.

Alternatively, by Green's theorem the integral inside the ellipse is

$$\iint_E g(x, y) \, dx \, dy = \int_{\partial E} G(x, y) \, dy$$

where $\frac{\partial G}{\partial x} = g$.

3.2 One side intersection, no vertex intersections

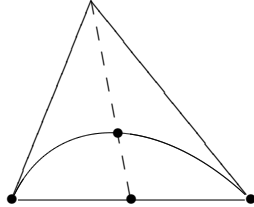
This case is entirely similar to the previous one.

3.3 One vertex intersection

This case is even simpler: T is contained in $\{f \geq 0\}$ or $\{f \leq 0\}$, we evaluate the sign of f at some interior point of the triangle in order to decide, and the integral will be that on the full triangle or zero.

3.4 Two vertex intersections

This case is more interesting. Either T is contained in one of the regions $\{f \geq 0\}$, $\{f \leq 0\}$, or it is divided in two regions by the conic. This can be discerned in the following way: determine a segment which cuts the triangle in two (not necessarily triangular) pieces, separating the two relevant vertices, and count the number of intersections of that segment and the conic. Examples: the median of the side determined by the two vertices, or a suitable vertical or horizontal segment. If there are intersections, we are in the latter situation, otherwise evaluate the sign of f inside the triangle to separate the first two possibilities.



Alternatively, convert the conic to a standard conic and check where the points lie after the transformation (see details in the next subsection).

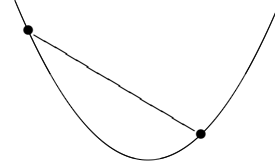
If the triangle is divided in two regions by the curve, we really have to compute the integral on a region bounded by a conic arc and one or two segments. As usual we can consider only the former (the bottom region in the above picture), without loss of generality. How to determine the actual region of integration? The sign of f in the bottom region is the same as the sign of f in the middle point of the bottom side, for example.

We can calculate the integral by affinely transforming the conic into a standard conic: the circle $x^2 + y^2 = 1$, the parabola $y = x^2$ or the hyperbola $xy = 1$.

1. Circle: the integral on the circular segment can be efficiently calculated as the integral on the circular sector

minus the integral on the triangle determined by the segment and the center of the circle.

2. Parabola: the integral after the transform is that on the region $\{y \in [l_{AB}(x), x^2], x \in [a_1, b_1]\}$ where (a_1, a_2) and (b_1, b_2) are the images of the two intersection vertices, with $a_1 < b_1$, and $l_{AB}(x)$ is the equation of the line through them.



3. Hyperbola: similarly to the previous case, the integral can be calculated as that on the region $\{y \in [l_{AB}(x), 1/x], x \in [a_1, b_1]\}$ if $a_1 < b_1 < 0$, or $\{y \in [1/x, l_{AB}(x)], x \in [a_1, b_1]\}$ if $0 < a_1 < b_1$.

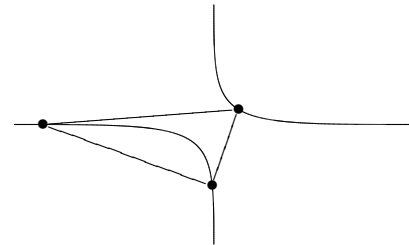
3.5 Three vertex intersections

As in the previous case, either T is contained in one of the regions $\{f \geq 0\}$, $\{f \leq 0\}$, or it is divided in two regions by the conic. This time we use a different method to differentiate the three possibilities, since in the third one we also need to know which are the two vertices through which the conic enters the triangle.

Since ellipses and parabolas define a convex region, a triangle with three vertices on such a curve cannot be divided by it. Thus, if the curve is of one of those types, it suffices once more to evaluate the sign of f in the triangle, and calculate the full triangle integral or return zero.

If $f = 0$ is a hyperbola, transform $f = 0$ into $xy = 1$. This curve defines two convex regions, limited by the branches $xy = 1, x < 0$ and $xy = 1, x > 0$. By inspecting the signs of the x -coordinates of the (transformed) vertices, we can determine in which branch they are.

1. If the three vertices are on the same branch of the hyperbola, the triangle is contained in $\{f \geq 0\}$ or $\{f \leq 0\}$, just determine the sign of f inside.
2. Otherwise, two vertices lie on one branch and the third vertex lies on the other branch. The integral is calculated as at the end of Section 3.4.



Note that the approach used in this case, namely the conversion to a standard conic in order to locate the vertices in relation to the curve, would have worked as well in Section 3.4, when we wanted to decide if the conic separates the triangle in two regions. This would amount to:

1. Ellipse: convert to $x^2 + y^2 = 1$ and decide if the third vertex is inside or outside the unit circle.

2. Parabola: convert to $y = x^2$ and decide if the third vertex is above or below the parabola.
3. Hyperbola: convert to $xy = 1$. If the two intersection vertices have different signs in their x -coordinates, the curve cannot separate the triangle. Otherwise, decide if the third vertex is in the convex region limited by the branch where the other two vertices are.

4. THE ALGORITHM

Algorithm 4.1 (next page) is a compilation of the steps described in the previous sections, so as to present an overview of the complete algorithm. Some case-by-case methods have not been explicitly written for brevity reasons.

4.1 Practical considerations

In relation to our implementation of this algorithm in MATLAB (almost complete as of May 2010) we would like to comment on numerical aspects that are not considered in our discussion above. First, several transformations suggested (Example 1.2 and the various transformations into standard conics from Section 3) are a source of rounding errors because for small regions the scaling needed is very large. This problem can be solved by avoiding all scalings, i.e. restricting the transformations to rotations and translations, not to a particular standard conic but to a member of some family of them. The result is a slight complication in the integration formulas, but nothing of concern in terms of efficiency.

An additional problem is that in some cases (the calculation suggested in Section 3.4 for the ellipse; Section 2.1) the sought integral is calculated as the difference of two easy integrals which may be orders of magnitude larger than the target, requiring much more precision in order not to lose significant digits.

5. APPLICATION: AN OPTIMAL CONTROL PROBLEM

Many technical processes are described by partial differential equations. Here, it is important to optimize these processes. This leads to optimization problems in an infinite-dimensional setting. As an prototype, we consider the minimization of a convex and quadratic functional subject to a linear elliptic partial differential equation and inequality constraints on the control. Let us briefly introduce the optimal control problem we have in mind.

Let $\Omega \subset \mathbb{R}^2$ be a bounded domain with C^3 -boundary Γ . For brevity, we will use $\xi = (x, y)$ to denote points in $\mathbb{R}^2$. Let us introduce the following elliptic equation

$$\begin{aligned} -\nabla \cdot (D(\xi) \nabla u(\xi)) + c(\xi) u(\xi) &= \chi_{\Omega'} f(\xi) & \text{in } \Omega, \\ u(\xi) &= 0 & \text{on } \Gamma. \end{aligned} \quad (3)$$

Here, the control is denoted by f , while the solution u of this system is the corresponding state. Thanks to the assumptions below, for each control $f \in L^2(\Omega)$ there exists a unique response $u \in H_0^1(\Omega)$, which is a weak solution of equation (3), see e.g. [2, Sect. 5.8]. The control acts on a compact polygonal subset $\Omega' \subset \Omega$. Now, we consider the control problem of minimizing

$$J(f, u) = \frac{1}{2} \iint_{\Omega} (u(\xi) - u_d(\xi))^2 d\xi + \frac{\alpha}{2} \iint_{\Omega'} f^2(\xi) d\xi \quad (4)$$

over all $f \in L^2(\Omega)$ subject to the elliptic equation (3) and the control constraints

$$f_a \leq f(\xi) \leq f_b \quad \text{a.e. on } \Omega. \quad (5)$$

That means, we want find a control f whose response u minimizes the distance to some desired state u_d . Let us denote this optimal control problem (3)–(5) by (P). The set of admissible controls for (P) is given by

$$F_{ad} = \{f \in L^2(\Omega) : f_a \leq f \leq f_b \quad \text{a.e. on } \Omega\}.$$

5.1 Existence and regularity of solutions

Concerning the data of the state equation (3), we make the following smoothness assumption on the data.

ASSUMPTION 5.1. *The coefficients in the differential operator satisfy $D \in C^{1,1}(\bar{\Omega})$ and $c \in C^{0,1}(\bar{\Omega})$. Moreover, we assume that $D(x) \geq D_0 > 0$ and $c(x) \geq 0$ for all $x \in \bar{\Omega}$.*

In order to obtain existence of solutions to (P) as well as a-priori discretization error estimates, we take the following assumptions on the data of the optimization problem.

ASSUMPTION 5.2. *We have $\alpha > 0$, $u_d \in H^1(\Omega)$, and $f_a, f_b \in \mathbb{R}$ with $f_a \leq f_b$ a.e. on Ω .*

Due to convexity, the problem under consideration is uniquely solvable, with solution denoted by (u^*, f^*) . Moreover, the solution can be characterized by the following necessary optimality conditions. These conditions are also sufficient since the optimal control problem is convex, see e.g. [4, Ch. 2].

THEOREM 5.3. *Let f^* be the solution of (P) with associated state u^* . Then there exists an adjoint state $p^* \in H^1(\Omega)$ such that the adjoint equation*

$$\begin{aligned} -\nabla \cdot (D(\xi) \nabla p^*(\xi)) + c(\xi) p^*(\xi) &= (u^* - u_d)(\xi) & \text{in } \Omega, \\ p^*(\xi) &= 0 & \text{on } \Gamma \end{aligned} \quad (6)$$

and the variational inequality

$$\iint_{\Omega'} (\alpha f^*(\xi) + p^*(\xi))(f(\xi) - f^*(\xi)) d\xi \geq 0 \quad \forall f \in F_{ad} \quad (7)$$

are satisfied. Moreover, the following pointwise representation of the optimal control holds

$$f^*(\xi) = \mathcal{P}_{[f_a, f_b]} \left(-\frac{1}{\alpha} p^*(\xi) \right) \quad \text{a.e. on } \Omega'. \quad (8)$$

Here, $\mathcal{P}_{[f_a, f_b]}(f)$ denotes the projection of $f \in \mathbb{R}$ on the interval $[f_a, f_b]$.

Using the projection representation of the optimal control, we can conclude higher regularity of the solution:

THEOREM 5.4. *Under the smoothness assumptions 5.1 and 5.2, it holds $u^*, p^* \in H^3(\Omega)$, $f^* \in H^1(\Omega)$.*

PROOF. Since we have $p^* \in H^1(\Omega)$ by the previous theorem, the projection representation (8) implies that the optimal control has the same regularity $f^* \in H^1(\Omega)$. Then the right-hand sides of (3) and (6) are functions in $H^1(\Omega)$. Standard regularity results for elliptic partial differential equations, e.g. [2, Thm. 8.13], yield $u^*, p^* \in H^3(\Omega)$. $\square$

ALGORITHM 4.1. Integrate a polynomial $g(x, y)$ of degree 4 on the intersection of a triangle T and the region $\{f \geq 0\}$ determined by a quadratic polynomial $f(x, y)$.

1. If $C := \{f = 0\}$ is a degenerate conic, go to step 9.
2. Calculate the intersections of C with each side of T .
3. If all sides of T are not free, let $L := \{T_1, \dots, T_n\}$ be a list of free triangular pieces as in Lemma 2.4, and go to step 6.
4. Otherwise, use Lemma 2.7 or Lemma 2.8 to obtain a list $L := \{T_1, \dots, T_n\}$ of free or almost-free triangular pieces.
5. For each triangle in L , if it is not free, substitute it in the list by the free pieces provided by Lemma 2.9.
6. Determine the type of C .
7. Initialize $S = 0$. For each triangle T_i in L :
 - 7.1. Let Z_i be the intersection of the border of T_i and C .
 - 7.2. If $Z_i = \emptyset$ or one non-vertex point:
 - A. If C is an ellipse, use Algorithm 3.1 to know the relative position of C and T_i .
 - i. If C is contained in T_i , determine the sign of f inside the ellipse. Let I be the integral of g on the bounded region inside C , or its complementary with respect to the full triangle, as needed.
 - ii. In any other case, determine the sign of f inside T_i . If it is positive, let $I = \iint_{T_i} g$, otherwise let $I = 0$.
 - B. If C is not an ellipse, determine the sign of f inside T_i . If it is positive, let $I = \iint_{T_i} g$, otherwise let $I = 0$.
 - C. Add I to S .
 - 7.3. If Z_i is one vertex: determine the sign of f in T_i . If it is positive let $I = \iint_{T_i} g$, otherwise let $I = 0$. Add I to S .
 - 7.4. If Z_i is two vertices:
 - A. Calculate the number of intersections of C with the segment from the middle point of the two vertices to the third vertex.
 - B. If there are none, determine the sign of f inside T_i . If positive, let $I = \iint_{T_i} g$, otherwise let $I = 0$. Add I to S .
 - C. If there is one, determine which of the two regions is the correct one, by evaluating f in a suitable point.
 - i. If C is an ellipse, transform it into $x^2 + y^2 = 1$. Calculate the integral on the circular segment. Let I be equal to that value or its complementary with respect to the full triangle.
 - ii. If C is a parabola, transform it into $y = x^2$. Calculate the integral between the segment and the arc of parabola (the segment is always above). Let I be equal to that value or its complementary with respect to the full triangle.
 - iii. If C is a hyperbola, transform it into $xy = 1$. Calculate the integral between the segment and the arc of hyperbola (which one is above depends on which branch the vertices are in). Let I be equal to that value or its complementary with respect to the full triangle.
 - iv. Add I to S .
 - 7.5. If Z_i is three vertices:
 - A. If C is an ellipse or a parabola, determine the sign of f inside T_i . If it is positive, let $I = \iint_{T_i} g$, otherwise let $I = 0$.
 - B. If C is a hyperbola, transform it into $xy = 1$ and determine in which branch does each vertex lie.
 - i. All in one branch: determine the sign of f inside T_i . If it is positive, let $I = \iint_{T_i} g$, otherwise let $I = 0$.
 - ii. Two vertices A, B in one branch and the third vertex in the other branch: calculate the integral between the segment AB and the arc of hyperbola (which one is above depends on which branch the vertices are in). Determine the sign of f in the middle point of AB . If positive, let I be equal to the calculated integral; if negative, to its complementary with respect to the full triangle.
 - C. Add I to S .
8. Output S and stop.
9. Determine the type of degenerate conic.
 - 9.1. If C is empty, one point, or a double line, determine the general sign of f . If it is positive, let $S = \iint_T g$, otherwise let $S = 0$. Output S and stop.
 - 9.2. Otherwise, if C is two parallel lines, convert it to $x^2 - x = 0$; if C is two crossing lines, convert it to $xy = 0$.
 - 9.3. Determine the position of the vertices with respect to the lines by examining the coordinates of their images by the transformation.
 - A. If all three vertices are in one of the regions, determine the sign of f inside T . If it is positive, let $S = \iint_T g$, otherwise let $S = 0$. Output S and stop.
 - B. Otherwise, determine the region(s) of integration by evaluating the sign of f at some vertex not on the conic. Write the region of integration as a sum of triangles with ± 1 coefficients. Calculate the integral according to this. Output the result and stop. (A case by case method can be easily written.)

5.2 Discretization and error estimate

Now, we turn to the discretization of (P). To that end, let us introduce a family of quasi-uniform triangulations of Ω , denoted by $\{\mathcal{T}_h\}_{h>0}$. Each triangulation is assumed to exactly fit the boundary of Ω , such that $\bar{\Omega} = \cup_{T \in \mathcal{T}_h} T$. This implies that elements of $\mathcal{T}_h$ lying on the boundary are curved. We further assume that for each $T \in \mathcal{T}_h$ there is a mapping Φ_T mapping the standard simplex $\hat{T}$ to T . Moreover, we require that the intersection of every triangle $T \in \mathcal{T}_h$ with the boundary of the control domain Ω' is empty. That is, the boundary of Ω' in Ω is completely resolved by edges of triangles.

With a triangulation we associate the following space of functions

$$V_h = \{v \in C(\bar{\Omega}) : \Phi_T(v|_T) \in P_2(\hat{T}) \ \forall T \in \mathcal{T}_h\},$$

which implies that functions $v_h \in V_h$ are polynomials of degree 2 on each triangular element. Since Ω' is a compact subset of Ω , there is a mesh size $h_0 > 0$ such that all elements $T \in \mathcal{T}_h$ with $\Omega' \cap T \neq \emptyset$ are triangular. Hence, the above developed integration procedure can be applied for functions $v_h \in V_h$ with support in Ω' .

Then the discrete optimal control problem can be written as: minimize $J(u_h, f_h)$ subject to $u_h \in V_h$, $f_h \in F_{ad}$

$$\iint_{\Omega} (D\nabla u_h \nabla v_h + cu_h v_h) d\xi = \iint_{\Omega'} f_h v_h d\xi \quad \forall v_h \in V_h. \quad (9)$$

Note that we did not explicitly require f_h to be in a finite-dimensional subspace. Nevertheless, if (u_h^*, f_h^*) is a solution of the discrete problem, there exists a discrete adjoint state $p_h^* \in V_h$ satisfying

$$\iint_{\Omega} (D\nabla p_h^* \nabla v_h + cp_h^* v_h) d\xi = \iint_{\Omega} (u_h^* - u_d) v_h d\xi \quad \forall v_h \in V_h \quad (10)$$

and

$$f_h^* = \mathcal{P}_{[f_a, f_b]} \left(-\frac{1}{\alpha} p_h^* \right). \quad (11)$$

Due to this projection representation, the control is implicitly discretized as the truncation of a function from the finite-dimensional space V_h .

THEOREM 5.5. *Let (u_h^*, f_h^*, p_h^*) be the solution of the discretized optimality system (9)–(11). Then there is a constant $c > 0$ independent of the mesh size h such that*

$$\|f_h^* - f^*\|_{L^2(\Omega)} + \|u_h^* - u^*\|_{H^1(\Omega)} + \|p_h^* - p^*\|_{H^1(\Omega)} \leq c h^3.$$

PROOF. Due to the approximation results of [1, Ch. 5.4], we have that the Assumption 2.4 in [3] is satisfied with $Z = H^3(\Omega) \cap H_0^1(\Omega)$ and convergence order h^3 . Then the claim follows by a direct application of [3, Thm. 2.4]. $\square$

Known estimates for piecewise linear elements yield a convergence order of h^2 only, compare [3]. In the two-dimensional case, i.e. $\Omega \subset \mathbb{R}^2$, the number of unknowns $N = 2 \dim V_h$ in the discretized problem is proportional to h^{-2} . Hence, our result implies that the approximation error is proportional to $N^{-3/2}$, whereas the use of linear polynomials only reduces the error like N^{-1} . This clearly shows that for optimal control problems as considered here, the use of piecewise quadratic approximations is preferable.

5.3 Solution method

In order to substitute f_h in (10) by the projection (11), integrals

$$\iint_{\{-\alpha^{-1}p_h < f_a\}} f_a v_h d\xi, \quad \iint_{\{f_a \leq -\alpha^{-1}p_h \leq f_b\}} p_h v_h d\xi$$

have to be evaluated for piecewise quadratic polynomials $v_h \in V_h$. This means, any solution method for the discretized problem encounters the difficulties of integrating over regions bounded by triangles and conics.

The system consisting of the equation (9)–(11) can be solved by means of a semi-smooth Newton method, see e.g. [3]. Within each step of the method, the non-smooth equation (11) is replaced by a linearized version

$$\chi_{\{-\alpha^{-1}p_h^k \in [f_a, f_b]\}} \left(f_h - \frac{1}{\alpha} p_h \right) = 0 \quad \text{on } \Omega',$$

where p_h^k is the adjoint state given by the previous step, and χ_A denotes the characteristic function of a set A . Multiplying this equation by a test function $v_h \in V_h$ and integrating on Ω' , we obtain

$$\begin{aligned} 0 &= \iint_{\Omega' \cap \{-\alpha^{-1}p_h^k \in [f_a, f_b]\}} \left(f_h - \frac{1}{\alpha} p_h \right) v_h d\xi \\ &= \sum_{T \in \mathcal{T}_h, T \cap \Omega' \neq \emptyset} \iint_{T \cap \{-\alpha^{-1}p_h^k \in [f_a, f_b]\}} \left(f_h - \frac{1}{\alpha} p_h \right) v_h d\xi \end{aligned}$$

for all $v_h \in V_h$. Here, it is important to be able to evaluate the integrals

$$\iint_{T \cap \{-\alpha^{-1}p_h \in [f_a, f_b]\}} \mathcal{P}_{[f_a, f_b]} \left(-\frac{1}{\alpha} p_h \right) v_h d\xi$$

and

$$\iint_{T \cap \{-\alpha^{-1}p_h \in [f_a, f_b]\}} f_h v_h d\xi,$$

which can be transformed to the type in the previous sections.

6. ACKNOWLEDGMENTS

The authors would like to thank J. Schicho for his suggestions and the participants of the Rastenfeld workshop for their feedback.

7. REFERENCES

- [1] S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics*. Springer, New York, third edition, 2008.
- [2] D. Gilbarg and N. S. Trudinger. *Elliptic partial differential equations of second order*. Springer-Verlag, Berlin, 1983.
- [3] M. Hinze. A variational discretization concept in control constrained optimization: the linear-quadratic case. *J. Computational Optimization and Applications*, 30:45–63, 2005.
- [4] F. Tröltzsch. *Optimale Steuerung partieller Differentialgleichungen*. Vieweg, Wiesbaden, 2005.
- [5] R. J. Walker. *Algebraic curves*. Springer-Verlag, New York, 1978. Reprint of the 1950 edition.

Computing the Singularities of Rational Space Curves

Xiaoran Shi Falai Chen^{*}
Department of Mathematics
University of Science and Technology of China
Hefei, Anhui 230026, P.R. China

ABSTRACT

In this paper, we discuss the singularities of rational space curves. Two methods are provided to compute the singularities of arbitrary degree curves. These methods are a generalization of the paper (Chen, Wang and Liu. Computing singular points of plane rational curves. Journal of Symbolic Computation 43, 92-117, 2008), which are based on the μ -basis of the rational space curve and on random technique. The μ -basis induces a matrix M which contains all the information about the singularities including the parameter values corresponding to the singularities, multiplicities and infinitely near singularities. These information can be obtained by computing the Smith form of the matrix M . We compare our methods with previous approaches such as generalized resultants, and provide some examples to illustrate the effectiveness of our methods.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—*Algebraic algorithms*; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*

General Terms

Algorithms

Keywords

Rational space curve, μ -basis, singularities.

1. INTRODUCTION

The singularities of curves and surfaces provide a great deal of information about the geometry and topology of the curves and surfaces. Detecting and analyzing singularity is very useful in geometric modeling and computer graphics.

^{*}Corresponding author. Email: chenfl@ustc.edu.cn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

There is a long history in the study of singularities of rational planar curves (see, for example, Abhyankar(1990), Chen and Sederberg(2002), Chionh and Sederberg(2001), Perez-Diaz(2007), Peterson(1917), Walker(1950)). The technique is based on either resultants or directly solving a non-linear system of equations (with two variables). However, there is much less work on the computation of the singularities of non-planar curves. Park discussed the singularities of n -dimensional polynomial curves based on Groebner basis computation [12]. Rubio, et. al. extended the method in [12] to rational parametric curves by generalized resultants [15], while Wang, Jia and Goldman applied the technique of moving planes and μ -basis to compute the singularities of three dimensional space curves of low degree (less than or equal to 6) [19].

The concept of μ -basis was developed by Cox, Sederberg and Chen in a series of papers (Sederberg and Chen(1995), Cox, Sederberg and Chen(1998), Chen, Zheng and Sederberg(2001), Chen and Wang(2002), Chen and Wang (2003), Chen, Cox and Liu(2005)) to devise robust and efficient methods to compute the implicit equations of rational curves and surfaces. Later it is shown that μ -basis can be applied successfully in computing the singular points of a rational planar curve, including order and infinitely near singularities [6]. The basis idea is as follows. Let M be the Bezout resultant matrix derived from the μ -basis of a planar rational curve. Then the Smith form of the matrix M deduces a series of *singularity factors* which provide all the information about singularities, including multiplicities, inversion formulas and infinitely near singularities. In [10, 11], the authors converted the singularity computation problem into the intersection of two planar curves and provided a proof for a conjecture in [6] regarding singularity factors. The purpose of this paper is to extend the methods in [6, 10, 11] to compute the singularities of rational space curves of arbitrary degree based on random technique. Compared with previous work in [6, 10, 11, 19], the main contributions of this paper are as follows. First, the paper provides methods to compute the singularities of rational space curves of arbitrary degree, while previous work deals only with planar curves or space curves of low degree; Second, the paper provides a method to handle the resultants of three polynomials using random technique. Thus methods with planar curves can be applied; Thirdly, it is straightforward to generalize the algorithms in the paper to higher dimensional curves. Compared with the work in [12, 15], the methods in this paper reduces three or higher dimensional curves to planar curves, and do not have to deal with generalized resultants. Furthermore, the methods in this paper give a more subtle characterization of

singularities such as multiplicities, infinitely near singularities than the methods in [12, 15].

We proceed in the following fashion. In Section 2, we introduce the notion of singularities for rational space curves, and the blow-up method which can find all the infinitely near singular points. In Section 3, some preliminary knowledge about μ -bases of rational space curves is summarized. Section 4 devotes to the main results—computing the singularities of rational space curves using the Smith form of the Bezout matrices derived from the μ -basis. The basis idea is as follows. First we randomly generate two sets of polynomials F_i, G_i , $i = 1, 2$ by forming linear combination of the μ -basis. Let M_i be the Bezout resultant matrix of F_i, G_i , $i = 1, 2$. Then the GCD of the Smith forms of M_1 and M_2 gives the *singularity factors* of the rational space curve. From the singularity factors, the locations, multiplicities and infinitely near singularities can be computed. In Section 5, an algorithm based on projection method is provided to compute singularities. Comparisons are also made between the two methods and between the methods in this paper and generalized resultants. We conclude in Section 6 with some remarks and future research problems.

2. SINGULARITIES AND BLOW UPS

Let $\mathbb{R}[s, u]$ be the set of homogeneous polynomials in the homogeneous parameter $s : u$ with real coefficients. A degree n rational space curve is usually written in homogeneous form as

$$\mathbf{P}(s, u) = (a(s, u), b(s, u), c(s, u), d(s, u)) \quad (1)$$

where $a(s, u)$, $b(s, u)$, $c(s, u)$, $d(s, u)$ are degree n homogeneous polynomials in $\mathbb{R}[s, u]$. Throughout this paper, we will assume that the four polynomials $a(s, u)$, $b(s, u)$, $c(s, u)$, $d(s, u)$ are relatively prime and linearly independent. Furthermore, we shall assume that the parametrization of the rational space curve $\mathbf{P}(s, u)$ is generically one-to-one.

2.1 Singularities

A singular point is a point on a curve (surface) where the tangent line (plane) is not uniquely determined.

Definition 1. Let $\mathbf{Q}$ be a point on a rational space curve $\mathcal{C}$, and let Π be a plane containing $\mathbf{Q}$. Then the point $\mathbf{Q}$ is called a **singular point of order** $k \geq 2$, if the intersection multiplicity of $\mathcal{C}$ with Π at $\mathbf{Q}$ is $k \geq 2$ for every generic choice of Π .

Remark 1. Let $\mathbf{Q}$ be a singular point of order $k \geq 2$ on a space curve $\mathcal{C}$ given by a rational parametrization $\mathbf{P}(s, u)$ as in (1). Let $\Pi := a_0x + b_0y + c_0z + d_0w = 0$ be a generic plane containing $\mathbf{Q}$. Define $\Phi(s, u) := a_0a(s, u) + b_0b(s, u) + c_0c(s, u) + d_0d(s, u)$. $\Phi(s, u)$ contains a factor $h(s, u)$ which is independent of the choice of Π . The polynomial $h(s, u)$ has degree k , and the roots of which are parameters corresponding to $\mathbf{Q}$. We call $h(s, u)$ the **inversion formula** for the point $\mathbf{Q}$.

Example 1. Let a rational space curve be given by

$$\mathbf{P}(s, u) = (s^9, s^5u^4, s^2u^7, u^9),$$

and let $a_0x + b_0y + c_0z = 0$ be a generic plane containing $\mathbf{Q}_1 = (0, 0, 0, 1)$. Then $\Phi(s, u) = a_0s^9 + b_0s^5u^4 + c_0s^2u^7 =$

$s^2(a_0s^7 + b_0s^3u^4 + c_0u^7)$. So $\mathbf{Q}_1$ is a double point with the inversion formula $h(s, u) = s^2$. Similarly, $\mathbf{Q}_2 = (1, 0, 0, 0)$ is a 4th order singularity with the inversion formula $h(s, u) = u^4$.

2.2 Blow ups

Given a rational space curve $\mathbf{P}(s, u)$, if a singular point $\mathbf{Q}$ is non-ordinary, there may be additional singularities $\mathbf{Q}^*$ arising from the singular point $\mathbf{Q}$ when the space curve $\mathbf{P}(s, u)$ undergoes a small perturbation. We call the singularities $\mathbf{Q}^*$ **infinitely near points** arising from the point $\mathbf{Q}$.

In general, we can move $\mathbf{Q}$ to the origin $(0, 0, 0, 1)$. Then the parametrization of the curve $\mathbf{P}(s, u)$ becomes

$$\mathbf{P}(s, u) = (a(s, u)h(s, u), b(s, u)h(s, u), c(s, u)h(s, u), d(s, u)), \quad (2)$$

where $\gcd(a, b, c) = 1$, $\gcd(h, d) = 1$, and $h(s, u)$ is the inversion formula of $\mathbf{Q}$. We can also make sure that $\gcd(a, h) = 1$ by a coordinate transformation.

Given a rational space curve in the form of (2), infinitely near singularities to a singular point $\mathbf{Q} = (0, 0, 0, 1)$ can be found by blowing up the rational space curve at $\mathbf{Q}$. Let $\mathbf{P}^1(s, u)$ be the homogeneous form of the curve

$$\left(\frac{x}{w}, \frac{y}{x}, \frac{z}{x}\right) = \left(\frac{ah}{d}, \frac{b}{a}, \frac{c}{a}\right),$$

so

$$\mathbf{P}^1(s, u) = (a^2h, bd, cd, ad). \quad (3)$$

A point $\mathbf{Q}^*$ is an infinitely near singularity in the first neighborhood of $\mathbf{Q}$ if $\mathbf{Q}^*$ is a singularity on the blow up curve $\mathbf{P}^1(s, u)$ and $\mathbf{Q}^*$ is related to $\mathbf{Q}$, i.e., all the parameters of $\mathbf{Q}^*$ form a subset of all the parameters corresponding to $\mathbf{Q}$. If we continue to blow up the space curve $\mathbf{P}^1(s, u)$ to get $\mathbf{P}^2(s, u)$, the points on the curve $\mathbf{P}^2(s, u)$ related to the point $\mathbf{Q}$ are said to be in the second neighborhood of $\mathbf{Q}$, and so on. Thus we have the following definition

Definition 2. We say that there is an infinitely near singular point of multiplicity r arising from the i -th neighborhood of the point $\mathbf{Q}$, if there is a singularity $\mathbf{Q}^*$ of multiplicity r on the i -th blow up space curve $\mathbf{P}^i(s, u)$, whose corresponding parameters are parameters corresponding to the point $\mathbf{Q}$.

Example 2. Reconsider *Example 1*. $\mathbf{Q}_1 = (0, 0, 0, 1)$ is a double singularity, and $\mathbf{Q}_2 = (1, 0, 0, 0)$ is a 4th order singularity. First we change the curve as form (2) at the point $\mathbf{Q}_1$:

$$\mathbf{P}_{\mathbf{Q}_1}(s, u) = (s^2u^7, s^5u^4, s^9, u^9).$$

To check if there are any infinitely near singularities arising from point $\mathbf{Q}_1$, we blow up the curve at point $\mathbf{Q}_1$:

$$\mathbf{P}_{\mathbf{Q}_1}^1(s, u) = (s^2u^5, s^3u^4, s^7, u^7),$$

we see that the point $\mathbf{Q}_1^* = (0, 0, 0, 1)$ is a double point related to the point $\mathbf{Q}_1$ on the curve $\mathbf{P}_{\mathbf{Q}_1}^1(s, u)$. So $\mathbf{Q}_1^*$ is an infinitely near double point in the first neighborhood of $\mathbf{Q}_1$.

Next we blow up the curve $\mathbf{P}_{\mathbf{Q}_1}^1(s, u)$ at the point $\mathbf{Q}_1^*$ and get

$$\mathbf{P}_{\mathbf{Q}_1}^2(s, u) = (s^2u^3, su^4, s^5, u^5).$$

There are no singularities on the curve $\mathbf{P}_{\mathbf{Q}_1}^2(s, u)$ related to $\mathbf{Q}_1^*$. We can similarly derive the infinitely near singularities arising from the singular point $\mathbf{Q}_2 = (1, 0, 0, 0)$.

$(0, 0, 0, 1) : \text{order 2 point} \rightarrow \text{order 2 point} \rightarrow \text{simple point}$

$(1, 0, 0, 0) : \text{order 4 point} \rightarrow \text{order 3 point} \rightarrow \text{simple point}$

3. μ -BASES OF RATIONAL SPACE CURVES

In this section, we review some basic knowledge about the μ -bases of rational space curves.

A **moving plane**

$$L(s, u; x, y, z, w) := A(s, u)x + B(s, u)y + C(s, u)z + D(s, u)w = 0 \quad (4)$$

is a set of planes with each homogeneous parameter $s : u$ corresponding to a plane, where $A(s, u)$, $B(s, u)$, $C(s, u)$ and $D(s, u)$ are homogeneous polynomials in $\mathbb{R}[s, u]$. We shall also write a moving plane in vector form $\mathbf{L}(s, u) := (A(s, u), B(s, u), C(s, u), D(s, u))$.

A moving plane $\mathbf{L}(s, u)$ is said to **follow** a rational space curve (1) if and only if

$$\mathbf{L}(s, u) \cdot \mathbf{P}(s, u) = aA + bB + cC + dD \equiv 0. \quad (5)$$

That is, for every homogeneous parameter $s_0 : u_0$, the plane

$$L(s_0, u_0; x, y, z, w) = A(s_0, u_0)x + B(s_0, u_0)y + C(s_0, u_0)z + D(s_0, u_0)w = 0$$

passes through the point $\mathbf{P}(s_0, u_0)$ on the curve $\mathbf{P}(s, u)$.

Let M_p be the set of all the moving planes following the rational space curve $\mathbf{P}(s, u)$, then M_p is a free syzygy module of rank three [17].

Definition 3. [17] The moving planes $\mathbf{p}(s, u), \mathbf{q}(s, u)$ and $\mathbf{r}(s, u)$ are called a μ -basis for the rational space curve $\mathbf{P}(s, u)$ if

1. $\mathbf{p}(s, u)$, $\mathbf{q}(s, u)$ and $\mathbf{r}(s, u)$ form a basis for the syzygy M_p , i.e., any moving plane $\mathbf{L}(s, u) \in M_p$ can be write as

$$\mathbf{L}(s, u) = \alpha(s, u)\mathbf{p}(s, u) + \beta(s, u)\mathbf{q}(s, u) + \gamma(s, u)\mathbf{r}(s, u),$$

where $\alpha(s, u), \beta(s, u), \gamma(s, u) \in \mathbb{R}[s, u]$.

2. $\deg(\mathbf{p}) + \deg(\mathbf{q}) + \deg(\mathbf{r}) = n$, where n is the degree of the curve $\mathbf{P}(s, u)$.

μ -basis has the following properties.

PROPOSITION 1. [17]

1. $[\mathbf{p}(s, u), \mathbf{q}(s, u), \mathbf{r}(s, u)] = \kappa \mathbf{P}(s, u)$, where κ is a nonzero constant, and $[\mathbf{p}, \mathbf{q}, \mathbf{r}]$ is the outer product of $\mathbf{p}, \mathbf{q}, \mathbf{r}$.
2. $\mathbf{p}(s, u), \mathbf{q}(s, u), \mathbf{r}(s, u)$ are linearly independent for every parameter (s, u) .

Write $\deg(\mathbf{p}(s, u)) = \mu_1$, $\deg(\mathbf{q}(s, u)) = \mu_2$, $\deg(\mathbf{r}(s, u)) = \mu_3$, and we can reorder $\mathbf{p}, \mathbf{q}, \mathbf{r}$ so that $\mu_1 \leq \mu_2 \leq \mu_3$. The μ -basis elements $\mathbf{p}, \mathbf{q}, \mathbf{r}$ for a rational space curve are not unique. But the degrees of the μ -basis elements μ_1, μ_2, μ_3 are unique [17]. We call (μ_1, μ_2, μ_3) the **type** of the μ -basis. Every rational space curve has a μ -basis. Moreover, there is a fast algorithm for computing a μ -basis based on Gaussian elimination [17].

4. COMPUTING SINGULARITIES WITH SMITH FORMS

In this section, we shall extend the method in [6, 10, 11] to compute the singularities of a rational space curve based on the Smith forms of matrices derived from μ -basis of the curve. We first make some preparations.

LEMMA 1. [19] Let $\mathbf{p}(s, u), \mathbf{q}(s, u), \mathbf{r}(s, u)$ be a μ -basis for the rational space curve $\mathbf{P}(s, u)$. Then the inversion formula for a point $\mathbf{Q}$ on the curve $\mathbf{P}(s, u)$ is given by the polynomial

$$h_{\mathbf{Q}} := \gcd(\mathbf{p}(s, u) \cdot \mathbf{Q}, \mathbf{q}(s, u) \cdot \mathbf{Q}, \mathbf{r}(s, u) \cdot \mathbf{Q}). \quad (6)$$

LEMMA 2. Let

$$\begin{aligned} \mathbf{h}_1(s, u) &= h_{11}u^{\mu_2-\mu_1}\mathbf{p} + h_{12}\mathbf{q}, \\ \mathbf{h}_2(s, u) &= h_{21}u^{\mu_3-\mu_1}\mathbf{p} + h_{22}u^{\mu_3-\mu_2}\mathbf{q} + h_{23}\mathbf{r} \end{aligned}$$

and

$$\begin{aligned} \mathbf{h}'_1(s, u) &= \mathbf{p}(s, u), \\ \mathbf{h}'_2(s, u) &= h'_{21}u^{\mu_3-\mu_1}\mathbf{p} + h'_{22}u^{\mu_3-\mu_2}\mathbf{q} + h'_{23}\mathbf{r} \end{aligned}$$

where $h_{11}, h_{12}, h_{21}, h_{22}, h_{23}, h'_{21}, h'_{22}, h'_{23} \in \mathbb{R}$ are nonzero random numbers, so $\mathbf{h}_1(s, u), \mathbf{h}_2(s, u), \mathbf{h}'_1(s, u), \mathbf{h}'_2(s, u)$ are homogeneous polynomial vectors. Then

$$\gcd(\mathbf{h}_1(s, u) \cdot \mathbf{Q}, \mathbf{h}_2(s, u) \cdot \mathbf{Q}) = h_{\mathbf{Q}}(s, u), \quad (7)$$

$$\text{or } \gcd(\mathbf{h}'_1(s, u) \cdot \mathbf{Q}, \mathbf{h}'_2(s, u) \cdot \mathbf{Q}) = h_{\mathbf{Q}}(s, u). \quad (8)$$

PROOF. Let $g_{\mathbf{Q}} = \gcd(\mathbf{h}_1(s, u) \cdot \mathbf{Q}, \mathbf{h}_2(s, u) \cdot \mathbf{Q})$. If $\mathbf{p} \cdot \mathbf{Q} = 0$, then

$$\begin{aligned} g_{\mathbf{Q}}(s, u) &= \gcd(h_{12}\mathbf{q} \cdot \mathbf{Q}, h_{22}u^{\mu_3-\mu_2}\mathbf{q} \cdot \mathbf{Q} + h_{23}\mathbf{r} \cdot \mathbf{Q}) \\ &= \gcd(\mathbf{q} \cdot \mathbf{Q}, \mathbf{r} \cdot \mathbf{Q}) = h_{\mathbf{Q}}(s, u). \end{aligned}$$

In the following, we assume $\mathbf{p} \cdot \mathbf{Q} \neq 0$, and

$$\mathbf{p} \cdot \mathbf{Q} = u^\alpha f_1(s, u), \quad \mathbf{q} \cdot \mathbf{Q} = u^\beta f_2(s, u), \quad \mathbf{r} \cdot \mathbf{Q} = u^\gamma f_3(s, u),$$

where $f_i(s, u)$ are homogenous polynomials which do not contain factor u , $i = 1, 2, 3$, and α, β, γ are nonnegative integers. If $\mathbf{q} \cdot \mathbf{Q} = 0$, set $\beta = +\infty$. Similarly, if $\mathbf{r} \cdot \mathbf{Q} = 0$ set $\gamma = +\infty$. Then

$$h_{\mathbf{Q}} = u^\delta \gcd(f_1, f_2, f_3),$$

where $\delta = \min(\alpha, \beta, \gamma)$. Similarly

$$g_{\mathbf{Q}} = u^{\delta'} \gcd(f_1, f_2, f_3)$$

for generic choice of coefficients h_{ij} . Here $\delta' = \min(\mu_2 - \mu_1 + \alpha, \beta, \gamma)$.

Now we consider two cases.

1. If $\alpha \geq \min(\beta, \gamma)$ or $\mu_1 = \mu_2$, then $\delta' = \delta$. Therefore $g_{\mathbf{Q}} = h_{\mathbf{Q}}$.
2. If $\alpha < \min(\beta, \gamma)$ and $\mu_1 < \mu_2$, then $\delta = \alpha$. In this case, $g_{\mathbf{Q}} = u^\lambda h_{\mathbf{Q}}$, where $\lambda = \delta' - \alpha > 0$.

Similarly, if we define

$$g'_{\mathbf{Q}} := \gcd(\mathbf{h}'_1(s, u) \cdot \mathbf{Q}, \mathbf{h}'_2(s, u) \cdot \mathbf{Q}),$$

then

$$g'_{\mathbf{Q}} = u^{\bar{\delta}} \gcd(f_1, f_2, f_3)$$

for generic choice of coefficients h'_{ij} . Here $\bar{\delta} = \min(\alpha, \mu_3 - \mu_2 + \beta, \gamma)$. Again we consider two cases.

1. If $\mu_2 = \mu_3$ or $\beta \geq \min(\alpha, \gamma)$, then $\bar{\delta} = \delta$. Therefore $g'_{\mathbf{Q}} = h_{\mathbf{Q}}$.
2. If $\beta < \min(\alpha, \gamma)$ and $\mu_2 < \mu_3$, then $g'_{\mathbf{Q}} = u^{\bar{\lambda}} h_{\mathbf{Q}}$ where $\bar{\lambda} = \bar{\delta} - \delta$.

Thus we either have $g_{\mathbf{Q}}(s, u) = h_{\mathbf{Q}}(s, u)$ or $g'_{\mathbf{Q}}(s, u) = h_{\mathbf{Q}}(s, u)$. This completes the proof of the lemma. $\square$

The above lemma makes us to handle two univariate polynomials (instead of three polynomials) whose GCD is the inversion formula like planar curve case. Thus we can make use of the Bezout matrix of the two polynomials and compute the Smith form of the matrix to obtain all the information about the singularities of the rational space curve $\mathbf{P}(s, u)$.

LEMMA 3. [6] *Let $f(t, u)$ and $g(t, u)$ be two homogeneous polynomials with degree m and n ($n \geq m$) respectively. Let $\text{Bez}(f, g)$ denote the Bezout resultant matrix of f and g . Then f and g have a greatest common divisor of degree r if and only if $\text{rank}(\text{Bez}(f, g)) = n - r$.*

Let $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}'_1, \mathbf{h}'_2$ be the polynomial vectors defined in Lemma 2. Let

$$\hat{B}(x, y, z, w) := \text{Bez}_{s,u}(\mathbf{h}_1 \cdot \mathbf{X}, \mathbf{h}_2 \cdot \mathbf{X}), \quad (9)$$

$$\hat{B}'(x, y, z, w) := \text{Bez}_{s,u}(\mathbf{h}'_1 \cdot \mathbf{X}, \mathbf{h}'_2 \cdot \mathbf{X}), \quad (10)$$

where $\mathbf{X} = (x, y, z, w)$.

THEOREM 1. *Let $\mathbf{Q} = (x_0, y_0, z_0, w_0)$ be a point on the curve $\mathbf{P}(s, u)$. Assume that $\text{rank}(\hat{B}(x_0, y_0, z_0, w_0)) = \mu_3 - r_1$ and $\text{rank}(\hat{B}'(x_0, y_0, z_0, w_0)) = \mu_3 - r_2$. Then the point $\mathbf{Q}$ is an order r singular point of $\mathbf{P}(s, u)$ if and only if $r = \min(r_1, r_2)$.*

PROOF. For a point $\mathbf{Q} = \mathbf{P}(s_0, u_0) = (x_0, y_0, z_0, w_0)$,

$$\hat{B}(x_0, y_0, z_0, w_0) = \text{Bez}_{s,u}(\mathbf{h}_1 \cdot \mathbf{Q}, \mathbf{h}_2 \cdot \mathbf{Q}).$$

Let $g_{\mathbf{Q}} = \gcd(\mathbf{h}_1 \cdot \mathbf{Q}, \mathbf{h}_2 \cdot \mathbf{Q})$. By Lemma 3, $r_1 = \deg(g_{\mathbf{Q}}) \geq \deg(h_{\mathbf{Q}})$. Here $h_{\mathbf{Q}}$ is the inversion formula for the point $\mathbf{Q}$. Similarly,

$$\hat{B}'(x_0, y_0, z_0, w_0) = \text{Bez}_{s,u}(\mathbf{h}'_1 \cdot \mathbf{Q}, \mathbf{h}'_2 \cdot \mathbf{Q}).$$

Let $g'_{\mathbf{Q}} = \gcd(\mathbf{h}'_1 \cdot \mathbf{Q}, \mathbf{h}'_2 \cdot \mathbf{Q})$, then $r_2 = \deg(g'_{\mathbf{Q}}) \geq \deg(h_{\mathbf{Q}})$. By Lemma 2

$$g_{\mathbf{Q}} = h_{\mathbf{Q}} \text{ or } g'_{\mathbf{Q}} = h_{\mathbf{Q}}.$$

Thus $\mathbf{Q}$ is an order r singular point if and only if $\deg(h_{\mathbf{Q}}) = r$, and if and only if $\min(r_1, r_2) = r$. $\square$

Define two Bezout matrices

$$\begin{aligned} B(s, u) &:= \text{Bez}_{t,v}(\mathbf{h}_1(t, v) \cdot \mathbf{P}(s, u), \mathbf{h}_2(t, v) \cdot \mathbf{P}(s, u)), \\ B'(s, u) &:= \text{Bez}_{t,v}(\mathbf{h}'_1(t, v) \cdot \mathbf{P}(s, u), \mathbf{h}'_2(t, v) \cdot \mathbf{P}(s, u)). \end{aligned} \quad (11)$$

The two matrices provide all the information about the singular points. To obtain such information, we resort to Smith forms of the two matrices.

Definition 4. Let $A(s, u)$ be an $m \times m$ matrix with entries in $\mathbb{R}[s, u]$. Suppose that the Smith form of the matrix $A(s, 1)$ is

$$\text{diag}(\bar{a}_m(s), \bar{a}_m(s)\bar{a}_{m-1}(s), \dots, \bar{a}_m(s) \cdots \bar{a}_1(s)),$$

and the Smith form of the matrix $A(1, u)$ is

$$\text{diag}(\hat{a}_m(u), \hat{a}_m(u)\hat{a}_{m-1}(u), \dots, \hat{a}_m(u) \cdots \hat{a}_1(u)).$$

Define $a_i(s, u) := \text{LCM}(\bar{a}_i(s, u), \hat{a}_i(s, u))$, where $\bar{a}_i(s, u)$ and $\hat{a}_i(s, u)$ are the homogenized polynomials of $\bar{a}_i(s)$ and $\hat{a}_i(u)$ respectively. Then the **Smith form** of the matrix $A(s, u)$ is given by

$$\begin{aligned} S(A(s, u)) &:= \\ \text{diag}(a_m(s, u), a_m(s, u)a_{m-1}(s, u), \dots, a_m(s, u) \cdots a_1(s, u)). \end{aligned} \quad (12)$$

Suppose that $S := S(B(s, u))$ and $S' := S(B'(s, u))$, define the **GCD** of the two diagonal matrices S and S' as a $\mu_3 \times \mu_3$ diagonal matrix $\gcd(S, S')$ whose diagonal elements are $\gcd(S[i, i], S'[i, i])$, $i = 1, \dots, \mu_3$.

THEOREM 2. *Let $S(s, u)$ and $S'(s, u)$ be defined as in Definition 4. Then*

$$\begin{aligned} \gcd(S, S') &= \\ \text{diag}(e_{\mu_3}(s, u), e_{\mu_3}(s, u)e_{\mu_3-1}(s, u), \dots, e_{\mu_3}(s, u) \cdots e_2(s, u), 0) \end{aligned} \quad (13)$$

and the zeros of $e_r(s, u)$ give all the parametric values of order r singular points (including infinitely near singular points). More precisely,

$$e_r(s, u) = h_r(s, u) \prod_{i \geq r} \psi_r^i(s, u) \quad (14)$$

where $h_r(s, u)$ is the product of the inversion formulas of all the order r singular points, and $\psi_r^i(s, u)$ is the inversion formula for all the order r infinitely near singularities in the neighborhood of order i ($i \geq r$) singular points on $\mathbf{P}(s, u)$.

PROOF. The first part of the theorem follows from Theorem 1. See the details in the reference [6]. The proof of the second part is similar to the proof in the reference [10]. $\square$

Definition 5. The homogenous polynomials $e_r(s, u)$, $r = 2, 3, \dots, \mu_3$ are called the **singularity factors** of the rational space curve $\mathbf{P}(s, u)$.

From (14), $e_r(s, u)$ may contain factors which correspond to singular points of order $> r$. Let $\tilde{e}_r(s, u)$ be the **modified singularity factor** by eliminating factors $\psi_r^i(s, u)$ ($i > r$) from $e_r(s, u)$, that is, $\tilde{e}_r(s, u) = h_r(s, u)\psi_r^r(s, u)$. $\tilde{e}_r(s, u)$ can be obtained by GCD computation from $e_r(s, u)$ [6].

Based on the above theorem, we provide an algorithm to compute singularities of rational space curves.

Algorithm 1: SINGULARITY-RSC-I

- Input: The parametric equation of a rational space curve $\mathbf{P}(s, u)$ as defined in (1).
- Output: Singularities of $\mathbf{P}(s, u)$ and their multiplicity.
- Algorithm:
 1. Compute a μ -basis $\mathbf{p}(s, u), \mathbf{q}(s, u), \mathbf{r}(s, u)$ for $\mathbf{P}(s, u)$.

2. Generate two polynomial vectors

$$\begin{aligned}\mathbf{h}_1(s, u) &= h_{11}u^{\mu_2-\mu_1}\mathbf{p}(s, u) + h_{12}\mathbf{q}(s, u) \\ \mathbf{h}_2(s, u) &= h_{21}u^{\mu_3-\mu_1}\mathbf{p}(s, u) + h_{22}u^{\mu_3-\mu_2}\mathbf{q}(s, u) \\ &\quad + h_{23}\mathbf{r}(s, u)\end{aligned}$$

where $h_{11}, h_{12}, h_{21}, h_{22}, h_{23}$ are random non-zero numbers. Compute the Bezout matrix $B(x, y, z, w)$ of $\mathbf{h}_1 \cdot \mathbf{X}$ and $\mathbf{h}_2 \cdot \mathbf{X}$ to obtain $B(s, u)$, and then compute the Smith form of $B(s, u)$. Here $\mathbf{X} = (x, y, z, w)$.

3. Generate two polynomial vectors

$$\begin{aligned}\mathbf{h}'_1(s, u) &= \mathbf{p}(s, u) \\ \mathbf{h}'_2(s, u) &= h'_{21}u^{\mu_3-\mu_1}\mathbf{p}(s, u) + h'_{22}u^{\mu_3-\mu_2}\mathbf{q}(s, u) \\ &\quad + h'_{23}\mathbf{r}(s, u)\end{aligned}$$

where $h'_{21}, h'_{22}, h'_{23}$ are random non-zero numbers. Compute the Bezout matrix $B'(x, y, z, w)$ of $\mathbf{h}'_1 \cdot \mathbf{X}$ and $\mathbf{h}'_2 \cdot \mathbf{X}$ to obtain $B'(s, u)$, and then compute the Smith form of $B'(s, u)$.

4. Compute

$$\begin{aligned}M(s, u) &= \gcd(S(B(s, u)), S(B'(s, u))) \\ &= \text{diag}(e_{\mu_3}(s, u), e_{\mu_3}(s, u)e_{\mu_3-1}(s, u) \\ &\quad, \dots, e_{\mu_3}(s, u) \cdots e_2(s, u), 0)\end{aligned}$$

5. For each $r = 2, 3, \dots, \mu_3$, compute the modified singularity factor $\tilde{e}_r$. Let

$$\begin{aligned}f_1(x, w) &= \text{Res}_{s,u}(d(s, u)x - a(s, u)w, \tilde{e}_r(s, u)), \\ f_2(y, w) &= \text{Res}_{s,u}(d(s, u)y - a(s, u)w, \tilde{e}_r(s, u)), \\ f_3(z, w) &= \text{Res}_{s,u}(d(s, u)z - a(s, u)w, \tilde{e}_r(s, u)).\end{aligned}$$

Let the zeros of $f_1(x, w) = 0$, $f_2(y, w) = 0$ and $f_3(z, w) = 0$ be x_i, y_j, z_k , $i, j, k = 1, 2, \dots, l$, respectively. Then each point (x_i, y_j, z_k) is a candidate of order r singularities. Check if it is indeed a singularities by substituting the point into the implicit equation of $\mathbf{P}(s, u)$ (which can be obtained by taking the resultants of the μ -basis of the curve). Output the singular points.

Remark 2. To identify if a singular point has infinitely near singularities, we just need to check if the singularity factor $e_r(s, u)$ contains factors coming from some $e_i(s, u)$ by computing $\gcd(e_r, e_i)$ ($i > r$). If the gcd is not 1, then some order i singularity contains some order r infinitely near singularity.

Remark 3. Theoretically the random numbers can be any nonzero real numbers. For our implementation, the coefficients of the parametric equation is integers (or rational numbers), and the random numbers are also chosen to be integers whose range is generally from -20 to 20 .

Remark 4. In the last step of the algorithm, if (x_i, y_j, z_k) is approximately computed, then substituting it into the implicit equation doesn't give exact zero. So here the criteria is that the absolute value of such substitution is less than some threshold.

We illustrate the algorithm with some examples.

Example 3. Consider the curve in *Example 1*:

$$\mathbf{P}(s, u) = (s^9, s^5u^4, s^2u^7, u^9).$$

A μ -basis is computed as

$$\begin{aligned}\mathbf{p}(s, u) &= (0, 0, u^2, -s^2) \\ \mathbf{q}(s, u) &= (0, u^3, -s^3, 0) \\ \mathbf{r}(s, u) &= (u^4, -s^4, 0, 0).\end{aligned}$$

Thus $\mathbf{P}(s, u)$ is a curve of type $(2, 3, 4)$. By generating two sets of random coefficients, we compute two Smith forms

$$S = \text{diag}(u^4, s^4u^7, s^4u^7\kappa_1, 0),$$

$$S' = \text{diag}(u^7, u^7, s^4u^{14}, 0),$$

where κ_1 is a polynomial of degree 14. Thus

$$\gcd(S, S') = \text{diag}(u^4, u^4u^3, u^7s^4, 0),$$

and $e_4(s, u) = u^4$, $e_3(s, u) = u^3$ and $e_2(s, u) = s^4$. Therefore $(s, u) = (1, 0)$ corresponds to a 4th order singular point $\mathbf{Q}_1 = (1, 0, 0, 0)$ which has an infinitely near triple point in its first neighborhood. $(s, u) = (0, 1)$ corresponds to a double point $\mathbf{Q}_2 = (0, 0, 0, 1)$ with an infinitely near double point in its first neighborhood.

Example 4. Given a rational space curve

$$\begin{aligned}\mathbf{P}(s, u) &= (s^{11}(s-u)^2(s-2u)^3, s^4(s-u)^4(s-2u)^5 \\ &\quad (-94u^3 - 55su^2 + 22s^2u - 7s^3), (s-u)^4 \\ &\quad (s-2u)^8(97u^4 - 62su^3 - 56s^3u + 87s^4), s^{16}).\end{aligned}$$

$\mathbf{P}(s, u)$ has a type $(5, 5, 6)$ whose μ -basis is too complicated to write down. We generate two sets of random coefficients and compute the GCD of Smith forms S and S' :

$$\begin{aligned}\gcd(S, S') &= \text{diag}(1, (u-s)^2(2u-s)^3, s^8(u-s)^4(2u-s)^5, \\ &\quad s^{11}(u-s)^4(2u-s)^5, s^{11}(u-s)^4(2u-s)^5, 0).\end{aligned}$$

Thus the singularity factors are $e_6 = 1$, $e_5 = (u-s)^2(2u-s)^3$, $e_4 = (u-s)^2(2u-s)^2s^8$, $e_3 = s^3$, $e_2 = 1$. So $\mathbf{Q}_1 = (0, 0, 0, 1)$ is a singular point of order 5 with the inversion formula $h_1 = (u-s)^2(2u-s)^3$. Blow up the point, there is an infinitely near point of order 4 with the inversion formula $(u-s)^2(2u-s)^2$. $\mathbf{Q}_2 = (1, 0, 0, 0)$ is a singular point of order 4 which contains an order 4 infinitely near singular point in the first neighborhood and an order 3 infinitely near singular point in the second neighborhood.

Example 5. Let a rational space curve be given by

$$\begin{aligned}\mathbf{P}(s, u) &= (s^{12}(s-u)^2(s-2u)^3(s-3u)^4(29u^4 + 44su^3 \\ &\quad + 87s^2u^2 - 23s^3u + 37s^4), s^5(s-u)^5(s-2u)^8 \\ &\quad (s-3u)^7, (s-u)^3(s-2u)^9(s-3u)^8(-29u^5 \\ &\quad - 8su^4 - 61s^2u^3 + 10s^3u^2 - 23s^4u + 98s^5), \\ &\quad s^{20}(-81u^5 + 40su^4 - 47s^2u^3 - 49s^3u^2 \\ &\quad + 11s^4u + 95s^5)).\end{aligned}$$

A μ -basis for $\mathbf{P}(s, u)$ is computed to show that $\mathbf{P}(s, u)$ has a type $(7, 9, 9)$. We generate two sets of random coefficients and compute the Smith matrices and their GCD:

$$\gcd(S, S') = \text{diag}(g_1, g_1, g_2, g_2, g_3, g_3, g_3, g_4, 0),$$

where

$$\begin{aligned} g_1 &= (-s+u)^2(2u-s)^3(3u-s)^4, \\ g_2 &= (-s+u)^3(2u-s)^6(3u-s)^7, \\ g_3 &= s^{10}(-s+u)^3(2u-s)^6(3u-s)^7, \\ g_4 &= s^{14}(-s+u)^4(2u-s)^8(3u-s)^8. \end{aligned}$$

Thus the singularity factors are $e_9 = (-s+u)^2(2u-s)^3(3u-s)^4$, $e_8 = 1$, $e_7 = (-s+u)(2u-s)^3(3u-s)^3$, $e_6 = 1$, $e_5 = s^{10}$, $e_4 = e_3 = 1$, $e_2 = s^4(-s+u)(2u-s)^2(3u-s)$. From the singularity factors, we know $\mathbf{P}(s, u)$ has two singular points: $\mathbf{Q}_1 = (0, 0, 0, 1)$ of order 9 and $\mathbf{Q}_2 = (1, 0, 0, 0)$ of order 5. Their resolutions are respectively

$$\begin{aligned} \mathbf{Q}_1 &: \text{order } 9 \rightarrow \text{order } 7 \rightarrow \text{two double points.} \\ \mathbf{Q}_2 &: \text{order } 5 \rightarrow \text{order } 5 \rightarrow \text{order } 2 \rightarrow \text{order } 2. \end{aligned}$$

5. COMPUTING SINGULARITIES BASED ON PROJECTION

Given a rational space curve $\mathbf{P}(s, u)$ as defined in (1), project the curve onto a random plane $L: Ax + By + Cz + Dw = 0$ to get a planar curve $\mathbf{P}'(s, u)$. For a generic choice of the projection plane, a singular point $\mathbf{Q}$ of $\mathbf{P}(s, u)$ maps to a singular point $\mathbf{Q}'$ of $\mathbf{P}'(s, u)$ with the same order, and both $\mathbf{Q}$ and $\mathbf{Q}'$ share the same inversion formula. However, it may happen that some ordinary points of $\mathbf{P}(s, u)$ map to a singular point of $\mathbf{P}'(s, u)$. To solve this problem, we pick up two random planes to get two different planar projection curves, and compute the common singularities. The common singularities should correspond to the singular points on the space curve $\mathbf{P}(s, u)$. To prove these observations, we first give a lemma.

LEMMA 4. Let $L := Ax + By + Cz + D = 0$ be a plane with nonzero coefficients A, B, C, D . Then the projection curve of a rational space curve $\mathbf{P}(s, u)$ onto the plane L is given by

$$\mathbf{P}'(s, u) = \mathbf{P}(s, t)T,$$

where T is a 4×4 matrix

$$T = \begin{pmatrix} B^2 + C^2 & -AB & -AC & 0 \\ -AB & A^2 + C^2 & -BC & 0 \\ -AC & -BC & A^2 + B^2 & 0 \\ -AD & -BD & -CD & A^2 + B^2 + C^2 \end{pmatrix}. \quad (15)$$

PROOF. For any point $\mathbf{P} = (x/w, y/w, z/w)$ in three dimensional space, let $\mathbf{P}' = (x'/w', y'/w', z'/w')$ be the projection of $\mathbf{P}$ onto the plane L . Then $\mathbf{PP}'$ is parallel to the vector (A, B, C) , so

$$\frac{x'/w' - x/w}{A} = \frac{y'/w' - y/w}{B} = \frac{z'/w' - z/w}{C}.$$

On the other hand, since $\mathbf{P}'$ is on the plane L ,

$$Ax' + By' + Cz' + Dw' = 0.$$

From the above two equations, we can get

$$(x', y', z', w') = (x, y, z, w)T.$$

This completes the proof. $\square$

THEOREM 3. Let $\mathbf{P}(s, u)$ be a rational space curve and $\mathbf{P}'(s, u)$ be the projection curve of $\mathbf{P}(s, u)$ onto a generic plane L . Then both $\mathbf{P}(s, u)$ and $\mathbf{P}'(s, u)$ have the same

generic singularities (including infinitely near singularities) under an invertible projective transformation, and the corresponding singularities share with the same inversion formulas.

PROOF. Let $\mathbf{Q}$ be a singular point of $\mathbf{P}(s, u)$ with order r , and $\mathbf{Q}'$ be the projection of $\mathbf{Q}$ onto the plane L . Without loss of generality, we assume $\mathbf{Q} = (0, 0, 0, 1)$ and

$$\mathbf{P}(s, u) = (a(s, u)h(s, u), b(s, u)h(s, u), c(s, u)h(s, u), d(s, u)),$$

where $\gcd(a, b, c) = 1$, $\gcd(a, h) = 1$, $\gcd(d, h) = 1$, and $h(s, u)$ is the inversion formula of $\mathbf{Q}$. By Lemma 4, $\mathbf{P}'(s, u) = \mathbf{P}(s, u)T$ and $\mathbf{Q}' = \mathbf{Q}T$. Here T is the matrix defined in (15).

We further assume $A^2 + B^2 + C^2 = 1$ (where A, B, C are all nonzero), then it is easy to verify that the matrix T in (15) is equivalent to

$$T' = \begin{pmatrix} 0 & -B & 0 & 0 \\ 0 & A & -C & 0 \\ 0 & 0 & B & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

that is, there exists an invertible matrix M such that $T' = TM$. Let

$$\bar{\mathbf{P}}(s, u) = \mathbf{P}(s, u)T' = (0, (Ab - Ba)h, (Bc - Cb)h, d).$$

Then $\bar{\mathbf{P}}(s, u) = \mathbf{P}'(s, u)M$, which implies that $\bar{\mathbf{P}}(s, u)$ and $\mathbf{P}'(s, u)$ have the same singularities (including infinitely near singularities) under an invertible projective transformation, and share with the same inversion formulas. Since $\gcd(a, b, c) = 1$, $\gcd(Ab - Ba, Bc - Cb) = 1$ for generic choice of A, B, C . Thus $\bar{\mathbf{P}}(s, u)$ has the singularity $\mathbf{Q} = (0, 0, 0, 1)$ of order r with the inversion formula $h(s, u)$. Therefore $\mathbf{P}'(s, u)$ has a corresponding singular point of the same order and the same inversion formula with $\mathbf{P}(s, u)$.

Blow up $\mathbf{P}(s, u)$ at $\mathbf{Q}$ to obtain the curve

$$\mathbf{P}_1(s, u) = (a^2h, bd, cd, ad).$$

Thus $\mathbf{P}(s, u)$ has an infinitely near singular point $\mathbf{Q}_1$ of order r_1 in the first neighborhood of $\mathbf{Q}$ if and only if $r_1 = \deg(h_1) > 1$ with the inversion formula $h_1 = \gcd(b, c, h)$. Similarly, $\bar{\mathbf{P}}(s, u)$ is blow up to

$$\bar{\mathbf{P}}_1(s, u) = (0, (Ab - Ba)^2h, (Bc - Cb)d, (Ab - Ba)d).$$

Thus $\bar{\mathbf{P}}(s, u)$ has a singular point $\bar{\mathbf{Q}}_1$ of order $\bar{r}_1$ in the infinitely near neighborhood of $\mathbf{Q}$ if and only if $\bar{r}_1 = \deg(\bar{h}_1) > 1$ with the inversion formula $\bar{h}_1 = \gcd(Bc - Cb, h)$. For generic choice of B, C , $\bar{h}_1 = h_1$, $\bar{r}_1 = r_1$ and $\bar{\mathbf{Q}}_1 = \mathbf{Q}_1$.

The above process can be repeated. Generally let $b_{i-1} = h_i b_i$, $c_{i-1} = h_i c_i$ with $b_0 = b$, $c_0 = c$, and $h_i = \gcd(b_{i-1}, c_{i-1}, h_{i-1})$ with $h_0 = h$. Then both $\mathbf{P}(s, u)$ and $\bar{\mathbf{P}}(s, u)$ have an infinitely near singular point $\mathbf{Q}_i$ of $\mathbf{Q}$ in the i -th neighborhood with the inversion formula h_i . That is, $\mathbf{P}(s, u)$ and $\bar{\mathbf{P}}(s, u)$ have the same singularities (including infinitely near singularities). Therefore $\mathbf{P}(s, u)$ and $\mathbf{P}'(s, u)$ have the same singularities under a projective transformation. The theorem is thus proved. $\square$

When a space curve $\mathbf{P}(s, u)$ is projected onto a plane L , the projection curve $\mathbf{P}'(s, u)$ can be regarded as a degenerated space curve. Let $\mathbf{p}'(s, u)$, $\mathbf{q}'(s, u)$, $\mathbf{r}'(s, u)$ be a μ -basis of $\mathbf{P}'(s, u)$. In this case, the μ -basis element $\mathbf{p}'(s, u)$ is a

constant vector (A, B, C, D) . Thus the inversion formula of a point $\mathbf{Q}'$ on $\mathbf{P}'(s, u)$ is

$$h_{\mathbf{Q}'} = \gcd(\mathbf{q}' \cdot \mathbf{Q}', \mathbf{r}' \cdot \mathbf{Q}').$$

Let

$$\hat{B}'(x, y, z, w) = \text{Bez}(\mathbf{q}' \cdot \mathbf{X}, \mathbf{r}' \cdot \mathbf{X}),$$

where $\mathbf{X} = (x, y, z, w)$, and

$$B'(s, u) = \hat{B}'(a'(s, u), b'(s, u), c'(s, u), d'(s, u)),$$

where (a', b', c', d') is the parametric equation of $\mathbf{P}'(s, u)$.

Compute the Smith form of $B'(s, u)$ to get the singularity factors of the projection curve $\mathbf{P}'(s, u)$. For another random plane, compute the singularity factors using the same approach. The GCD of the corresponding singularity factors provide information about the singularities for the space curve $\mathbf{P}(s, u)$.

In the following, we describe the algorithm to compute the singularities of rational space curves based on projection method.

Algorithm 2 SINGULARITY-RSC-II

- Input: The parametric equation of a rational space curve $\mathbf{P}(s, u)$ as defined in (1).
- Output: Singularities of $\mathbf{P}(s, u)$ and their multiplicity.
- Algorithm:
 1. Compute a μ -basis $\mathbf{p}(s, u), \mathbf{q}(s, u), \mathbf{r}(s, u)$ for $\mathbf{P}(s, u)$.
 2. Generate a random plane $L_1 := A_1x + B_1y + C_1z + D_1w = 0$. Compute the projection curve $\mathbf{P}_1(s, u)$ of $\mathbf{P}(s, u)$ on the plane L . Compute a μ -basis $\mathbf{p}_1(s, u), \mathbf{q}_1(s, u), \mathbf{r}_1(s, u)$ for $\mathbf{P}_1(s, u)$, and then compute $B_1(s, u) = \text{Bez}(\mathbf{q}_1 \cdot \mathbf{X}, \mathbf{r}_1 \cdot \mathbf{X})|_{\mathbf{X}=\mathbf{P}_1(s, u)}$ to obtain $S_1(s, u) = S(B_1(s, u))$.
 3. Generate another random plane L_2 , and do the same process with the above step to get $S_2(s, u)$.
 4. Compute the GCD of the singularity factors of the same order for $S_1(s, u)$ and $S_2(s, u)$ to get the singularity factors $e_r(s, u)$ of $\mathbf{P}(s, u)$, $r = 2, 3, \dots, l$.
 5. The rest of the steps is similar to those of algorithm SINGULARITY-RSC-I.

We provide two examples to illustrate the above algorithm.

Example 6. Consider a rational space curve

$$\mathbf{P}(s, u) = (u^2(s-u)^3(s-2u)^2, us(s-u)^3(s-3u)^2, u^3(s-2u)^2(s-3u)^2, (s-u)^3(s-2u)^2(s-3u)^2).$$

Generating two random planes, and project the curve $\mathbf{P}(s, u)$ onto the two planes. The two Smith matrices are computed

$$S = \text{diag}(1, (u-s)^3, (u-s)^3(3u-s)^2(2u-s)^2\kappa_1, 0),$$

$$S' = \text{diag}(1, (u-s)^3, (u-s)^3(3u-s)^2(2u-s)^2\kappa_2, 0),$$

where $\gcd(\kappa_1, \kappa_2) = 1$. The GCD of S and S' is

$$\gcd(S, S') = \text{diag}(1, (u-s)^3, (u-s)^3(3u-s)^2(2u-s)^2, 0).$$

So $(s, u) = (1, 1)$ corresponds to a triple point $(0, 0, 1, 0)$, and the inversion formula is $h = (u-s)^3$. $(s, u) = (3, 1)$ corresponds to a double point $(1, 0, 0, 0)$ with the inversion formula $h = (3u-s)^2$. $(s, u) = (2, 1)$ corresponds to another double point $(0, 1, 0, 0)$ whose inversion formula is $h = (2u-s)^2$.

The following example shows that the projection curve doesn't have the nice property as sated in *Theorem 3* if the condition $\gcd(a, h) = 1$ is not satisfied.

Example 7. Consider the same rational space curve as in *Example 1*:

$$\mathbf{P}(s, u) = (s^9, s^5u^4, s^2u^7, u^9).$$

Generating two random planes and project the curve onto the two planes. Compute the μ -bases and the Smith forms respectively. We get

$$S = \text{diag}(1, u^8, u^8, u^8s^4\kappa_1, 0), \quad S' = \text{diag}(1, u^8, u^8, u^8s^4\kappa_2, 0),$$

where $\gcd(\kappa_1, \kappa_2) = 1$. The GCD of the two Smith matrices is thus $\text{diag}(1, u^8, u^8, u^8s^4, 0)$. From the Smith form, we are not able to compute correct singularity factors and infinitely near singularities. The reason is that condition $\gcd(a, h) = 1$ for each singular point is not satisfied. To compute correct singularity factors, one can replace $\mathbf{P}(s, u)$ with an equivalent curve $\mathbf{P}(s, u)T$ such that condition $\gcd(a, h) = 1$ is satisfied, where T is a 4×4 invertible matrix. For two random planes, the Smith forms of the projection curves are computed, and their GCD is $\text{diag}(1, u^4, u^7, s^4u^7, 0)$, which gives the correct result.

We end this section with some computational comparisons of the two algorithms (RSC-I and RSC-II) in the paper, and the generalized resultant method in [15] to compute the singularities of space rational curves.

For the two algorithms RSC-I and RSC-II, table 1 lists the computational time for the three examples (*Examples 3-5*), including the time for computing μ -bases, Smith forms and total time, where — stands for time which is longer than one hour. Since the algorithm RSC-II has to deal with Bezout matrices of higher order because of higher degree of μ -bases, Smith form computation takes longer time. Thus algorithm RSC-I is faster than algorithm RSC-II.

Table 1: Comparisons of computational time (in seconds) for RSC-I and RSC-II

Examples	Algorithms	μ -bases	Smith form	total
Example 3	RSC-I	0.172	0.093	0.256
	RSC-II	0.203	0.188	0.391
Example 4	RSC-I	0.359	7.937	8.296
	RSC-II	0.701	135.0	135.7
Example 5	RSC-I	0.547	230.0	230.5
	RSC-II	0.875	—	—

For the generalized resultant method, the main computational cost is in the GCD (denoted by $g(t)$) computation of a large set of univariate polynomials. Table 2 lists the time to compute $g(t)$ for the generalized resultant method. Note that the generalized method is not able to detect multiplicities and infinitely near singularities from $g(t)$. In fact,

$g(t)$ is equivalent to the largest principle leading minor of the Bezout matrices in algorithms RSC-I and RSC-II. As a comparison, we also list the time to compute the largest leading principle minors of the Bezout matrices in RSC-I and RSC-II. Our methods are more advantageous than generalized resultant method as the dimension of the curve increases.

Table 2: Computational time (in seconds) for generalized resultant

Examples	RSC-I	RSC-II	generalized resultant
Example 3	0.202	0.390	0.311
Example 4	1.905	10.19	4.922
Example 5	129.6	614.0	475.0

6. CONCLUSIONS

In this paper, two methods are provided to compute the singularities of a rational space curve. The main idea is to convert three dimensional problems into two dimensional problems by random technique. Thus the method of μ -basis and Smith form can be applied to compute singularities including location, multiplicity and infinitely near singularities. As a comparison, previous methods such as generalized resultant is not able to provide such detailed characterization of singularities. Statistics also suggest that the first algorithm in the paper is superior than other methods such as generalized resultant method in computational cost. Furthermore, it is straightforward to generalize our methods to handle singularities of higher dimensional curves.

It is worthwhile to generalize the method to compute the singularities of rational parametric surfaces.

7. ACKNOWLEDGEMENTS

The authors would like to thank the referees' critical comments to improve the manuscript. This work is supported by NSF of China (No. 60873109, 60225002), One Hundred Talent Project supported by CAS and the 111 Project (No. B07033).

8. REFERENCES

- [1] S. ABHYANKAR. 1990. Algebraic geometry for scientists and engineers. *American Mathematical Society*.
- [2] F. CHEN, D. COX AND Y. LIU. 2005. The μ -basis and implicitization of a rational parametric surface. *Journal of Symbolic Computation* 39, 689-706.
- [3] F. CHEN AND T. SEDERBERG. 2002. A new implicit representation of a planar rational curve with high order singularity. *Computer Aided Geometric Design* 19, 151-167.
- [4] F. CHEN AND W. WANG. 2002. The μ -basis of a planar rational curve properties and computation. *Graphical Models* 64, 368-381.
- [5] F. CHEN AND W. WANG. 2003. Revisiting the μ -basis of a rational ruled surface. *Journal of Symbolic Computation* 36, 699-716.
- [6] F. CHEN, W. WANG AND Y. LIU. 2008. Computing singular points of plane rational curves. *Journal of Symbolic Computation* 43, 92-117.
- [7] F. CHEN, J. ZHENG AND T. SEDERBERG. 2001. The μ -basis of a rational ruled surface. *Computer Aided Geometric Design* 18, 61-72.
- [8] E. CHIONH AND T. SEDERBERG. 2001. On the minors of the implicitization b ezout matrix for a rational plane curve. *Computer Aided Geometric Design* 18, 21-36.
- [9] D. COX, T. SEDERBERG AND F. CHEN. 1998. The moving line ideal basis of planar rational curves. *Computer Aided Geometric Design* 15, 803-827.
- [10] X. JIA AND R. GOLDMAN. Using Smith forms and μ -bases to compute all the singularities of rational planar curves. *Submitted*.
- [11] X. JIA AND R. GOLDMAN. 2009. μ -bases and singularities of rational planar curves. *Computer Aided Geometric Design* 26, 970-988.
- [12] H. PARK. 2002. Effective computation of singularities of parametric affine curves. *Journal of Pure and Applied Algebra* 173, 49-58.
- [13] S. PEREZ-DIAZ. 2007. Computation of the singularities of parametric plane curves. *Journal of Symbolic Computation* 42, 835-857.
- [14] O. PETERSON. 1917. The double points of rational curves. *American Mathematical Monthly* 42, 376-379.
- [15] R. RUBIO, J.M. SERRADILLA AND M.P. VELEZ. 2009. Detecting real singularities of a space curve from a real rational parameterization. *Journal of Symbolic Computation* 44, 490-498.
- [16] T. SEDERBERG AND F. CHEN. 1995. Implicitization using moving curves and surfaces. In: *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM Press, 301-308.
- [17] N. SONG AND R. GOLDMAN. 2009. μ -bases for polynomial systems in one variable. *Computer Aided Geometric Design* 26, 217-230.
- [18] R. WALKER. 1950. Algebraic curves. *Princeton University Press*.
- [19] H. WANG, X. JIA AND R. GOLDMAN. 2009. Axial moving planes and singularities of rational space curves. *Computer Aided Geometric Design* 26, 300-316.

Solving Schubert Problems with Littlewood-Richardson Homotopies*

Frank Sottile
Department of Mathematics
Texas A&M University
College Station, TX 77843, USA
sottile@math.tamu.edu

Ravi Vakil
Department of Mathematics
Stanford University
Stanford, CA 94305, USA
vakil@math.stanford.edu

Jan Verschelde
Dept of Math, Stat, and CS
University of Illinois at Chicago
Chicago, IL 60607, USA
jan@math.uic.edu

ABSTRACT

We present a new numerical homotopy continuation algorithm for finding all solutions to Schubert problems on Grassmannians. This Littlewood-Richardson homotopy is based on Vakil's geometric proof of the Littlewood-Richardson rule. Its start solutions are given by linear equations and they are tracked through a sequence of homotopies encoded by certain checker configurations to find the solutions to a given Schubert problem. For generic Schubert problems the number of paths tracked is optimal. The Littlewood-Richardson homotopy algorithm is implemented using the path trackers of the software package PHCpack.

Categories and Subject Descriptors

G.1.5 [Roots of Nonlinear Equations]: Continuation (homotopy) methods; G.2.1 [Combinatorics]: Counting Problems

General Terms

Algorithms

Keywords

continuation, geometric Littlewood-Richardson rule, Grassmannian, homotopies, numerical Schubert calculus, path following, polynomial system, Schubert problems.

1. INTRODUCTION

The Schubert calculus is concerned with geometric problems of the form: Determine the k -dimensional linear subspaces of $\mathbb{C}^n$ that meet a collection of fixed linear subspaces in specified dimensions. For example, what are the three-dimensional linear subspaces of $\mathbb{C}^7$ that meet each of 12 general four-dimensional linear subspaces in at least a line?

*This material is based upon work supported by the National Science Foundation under Grants DMS-0538734, DMS-0915211, DMS-0801196 and DMS-0713018.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

(There are 462 [14].) The traditional goal is to count the number of solutions and the method of choice for this enumeration is the Littlewood-Richardson rule, which comes from combinatorics and representation theory [4]. Recently, Vakil gave a geometric proof of this rule [20] through explicit specializations organized by a combinatorial checkers game.

Interest has grown in computing the solutions to actual Schubert problems. One motivation has been the experimental study of reality in the Schubert calculus [6, 18, 19, 17]. A proof of Pieri's rule (a special case of the Littlewood-Richardson rule) using geometric specializations [16] led to the Pieri homotopy for solving special Schubert problems [7]. This was implemented and refined [8, 11, 22, 23, 24], and has been used to address a problem in pure mathematics [9]. Another motivation is the output pole placement problem in linear systems control [1, 2, 3, 13, 23].

We present the Littlewood-Richardson homotopy, which is a numerical homotopy algorithm for finding all solutions to any Schubert problem. It is based on the geometric Littlewood-Richardson rule [20] and it is optimal in that generically there are no extraneous paths to be tracked.

We describe Schubert problems and their equations in §2, and give a detailed example of the geometric Littlewood-Richardson rule in §3. We then explain the local structure of the Littlewood-Richardson homotopy in §4. The next three sections give more details on the local coordinates, the moving flag, and the checker configurations. In §8 we discuss the global structure of the Littlewood-Richardson homotopy and conclude in §9 with a brief description of our PHCpack [21] implementation and timings.

2. SCHUBERT PROBLEMS

A Schubert problem asks for the k -dimensional subspaces of $\mathbb{C}^n$ that satisfy certain Schubert conditions imposed by general flags. We explain this in concrete terms.

A point in $\mathbb{C}^n$ is represented by a $n \times 1$ column vector and a linear subspace as the column span of a matrix. A flag F is represented by an ordered basis $\mathbf{f}_1, \dots, \mathbf{f}_n$ of $\mathbb{C}^n$ that forms the columns of a matrix F . If we write F_i for the span of the first i columns of F , then a Schubert condition imposed by F is the condition on the k -plane X that

$$\dim(X \cap F_{\omega_i}) \geq i \quad \text{for } 1 \leq i \leq k, \quad (1)$$

where $\omega \in \mathbb{N}^k$ is a bracket; $1 \leq \omega_1 < \omega_2 < \dots < \omega_k \leq n$.

If we set $|\omega| = \sum_i n - k + i - \omega_i$, then a Schubert problem is a list $\omega^1, \omega^2, \dots, \omega^s$ of brackets such that

$$|\omega^1| + |\omega^2| + \dots + |\omega^s| = k(n - k). \quad (2)$$

For example, the Schubert problem of three-planes meeting 12 four-planes in $\mathbb{C}^7$ is given by 12 equal codimension one brackets and is written succinctly as $[4\ 6\ 7]^{12}$. The numerical condition (2) ensures that if $F^1, \dots, F^s$ are general, then there are finitely many k -planes that satisfy condition ω^i for flag F^i , for $i = 1, \dots, s$.

The set of k -planes X satisfying (1) is the Schubert variety $\Omega_\omega(F)$. This is a subvariety of the $k(n-k)$ -dimensional Grassmannian of k -planes in n -space. Thus solving a Schubert problem corresponds to determining the intersection of Schubert varieties with respect to various flags.

These geometric conditions are formulated as systems of polynomials by parameterizing an appropriate subset of the Grassmannian. For example, for $F \in \mathbb{C}^{6 \times 6}$, the Schubert variety $\Omega_{[2\ 4\ 6]}(F)$ contains

$$X = \begin{bmatrix} 1 & 0 & 0 \\ x_{21} & 1 & 0 \\ x_{31} & x_{32} & 1 \\ x_{41} & x_{42} & x_{43} \\ 0 & x_{52} & x_{53} \\ 0 & 0 & x_{63} \end{bmatrix} \quad \begin{aligned} \dim(X \cap F_2) &= 1 \\ \dim(X \cap F_4) &= 2 \\ \dim(X \cap F_6) &= 3 \end{aligned} \quad (3)$$

Expressed via conditions on the minors of $[X|F_i]$ this is a system of 13 polynomials in 9 variables.

The most elementary Schubert problem involves only two brackets, ω and τ with $|\omega| + |\tau| = k(n-k)$. If F and M are general flags and $\omega^\vee = [n+1-\omega_k \dots n+1-\omega_1]$, then

$$\Omega_\omega(F) \cap \Omega_\tau(M) = \begin{cases} \langle \mathbf{x}_1, \dots, \mathbf{x}_k \rangle & \text{if } \tau = \omega^\vee, \\ \emptyset & \text{otherwise,} \end{cases} \quad (4)$$

where $\mathbf{x}_i = F_{\omega_i} \cap M_{n+1-\omega_i}$, which is one-dimensional and thus solved by linear algebra. Such elementary Schubert problems are the start systems for the Littlewood-Richardson homotopy.

When $|\omega| + |\tau| < k(n-k)$, an intersection

$$\Omega_\omega(F) \cap \Omega_\tau(M) \quad (5)$$

of Schubert varieties for general flags F and M has positive dimension. This intersection is homologous to a union of Schubert varieties $\Omega_\sigma(F)$ for $|\sigma| = |\omega| + |\tau|$, each occurring with multiplicity the Littlewood-Richardson number $c_{\omega, \tau}^\sigma$. We write this formally as a sum,

$$\Omega_\omega(F) \cap \Omega_\tau(M) \sim \sum_\sigma c_{\omega, \tau}^\sigma \Omega_\sigma(F). \quad (6)$$

In the geometric Littlewood-Richardson rule [20], the flag M moves into special position with respect to the flag F . This changes the intersection (5), breaking it into components which are transformed into Schubert varieties $\Omega_\sigma(F)$. Then $c_{\omega, \tau}^\sigma$ is the number of different ways to arrive at $\Omega_\sigma(F)$.

The relative position of the flags F and M is represented via a configuration of n black checkers in a $n \times n$ board with no two in the same row or column. The dimension of $M_a \cap F_b$ is the number of checkers weakly northwest of the square (a, b) . This is illustrated in Figure 1. Each cell corresponds to a vector space, and the vector space of each cell contains the vector spaces of the cells weakly northwest of it.

All components of the specializations of (5) are represented by placements of k red¹ checkers on a board with n checkers representing the relative positions of the flags. The red checkers represent the position of a typical k -plane

¹Red checkers look grey when printed in black and white.

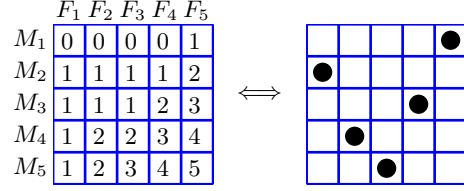


Figure 1: Dimension array $\dim M_a \cap F_b$ and corresponding checker configuration.

in the component as follows: If the k -plane meets the vector space corresponding to a cell in dimension ℓ , then there are ℓ red checkers weakly northwest of it. See Figure 2 for

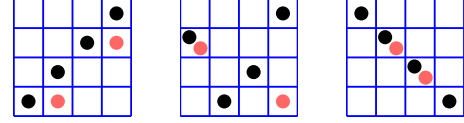


Figure 2: Three checkerboards with $n = 4$ and $k = 2$.

examples. We discuss the placement and movement of the checkers in §3 and §7.

Applying the geometric Littlewood-Richardson rule to two Schubert varieties in a Schubert problem of s brackets reduces it to Schubert problems involving $s-1$ brackets. The Littlewood-Richardson homotopy begins with the solutions to those smaller problems and reverses the specializations to solve the original Schubert problem.

Let $k = 3$ and $n = 6$, and consider this for the Schubert problem $[2\ 4\ 6]^3 = [2\ 4\ 6][2\ 4\ 6][2\ 4\ 6]$. Given three general flags F, M, N , we want to resolve the triple intersection

$$\Omega_{[2\ 4\ 6]}(F) \cap \Omega_{[2\ 4\ 6]}(M) \cap \Omega_{[2\ 4\ 6]}(N). \quad (7)$$

We first apply the geometric Littlewood-Richardson rule to the first intersection to obtain

$$(\Omega_{[2\ 3\ 4]}(F) + 2\Omega_{[1\ 3\ 5]}(F) + \Omega_{[1\ 2\ 6]}(F)) \cap \Omega_{[2\ 4\ 6]}(N), \quad (8)$$

and then apply (4) to obtain $2\langle \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \rangle$, where $\mathbf{x}_1 = F_1 \cap N_6$, $\mathbf{x}_2 = F_3 \cap N_4$, and $\mathbf{x}_3 = F_5 \cap N_2$.

The Littlewood-Richardson homotopy starts with the single 3-plane $\langle \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \rangle$ (counted twice) which is the unique solution to (8). It then numerically continues this solution backwards along the geometric specializations transforming (7) into (8) to arrive at solutions to (7). As the multiplicity 2 of $\Omega_{[1\ 3\ 5]}(F)$ in (8) is the number of paths in the specialization that end in $\Omega_{[1\ 3\ 5]}(F)$, the single solution $\langle \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \rangle$ that we began with yields two solutions to (7).

3. THE PROBLEM OF FOUR LINES

We illustrate the Littlewood-Richardson homotopy via the classical problem of which lines in projective three-space ($\mathbb{P}^3$) meet four given lines. This corresponds to two-planes in $\mathbb{C}^4$ meeting four fixed two-planes nontrivially, or $[2\ 4]^4$.

A flag in $\mathbb{P}^3$ consists of a point lying on a line that is contained in a plane (depicted here as a triangle). Figure 3 shows the specialization of two flags, one fixed and one moving, that underlies the geometric Littlewood-Richardson rule for every Schubert problem in $\mathbb{P}^3$. The top shows the geometry of the specialization. Below are matrices representing

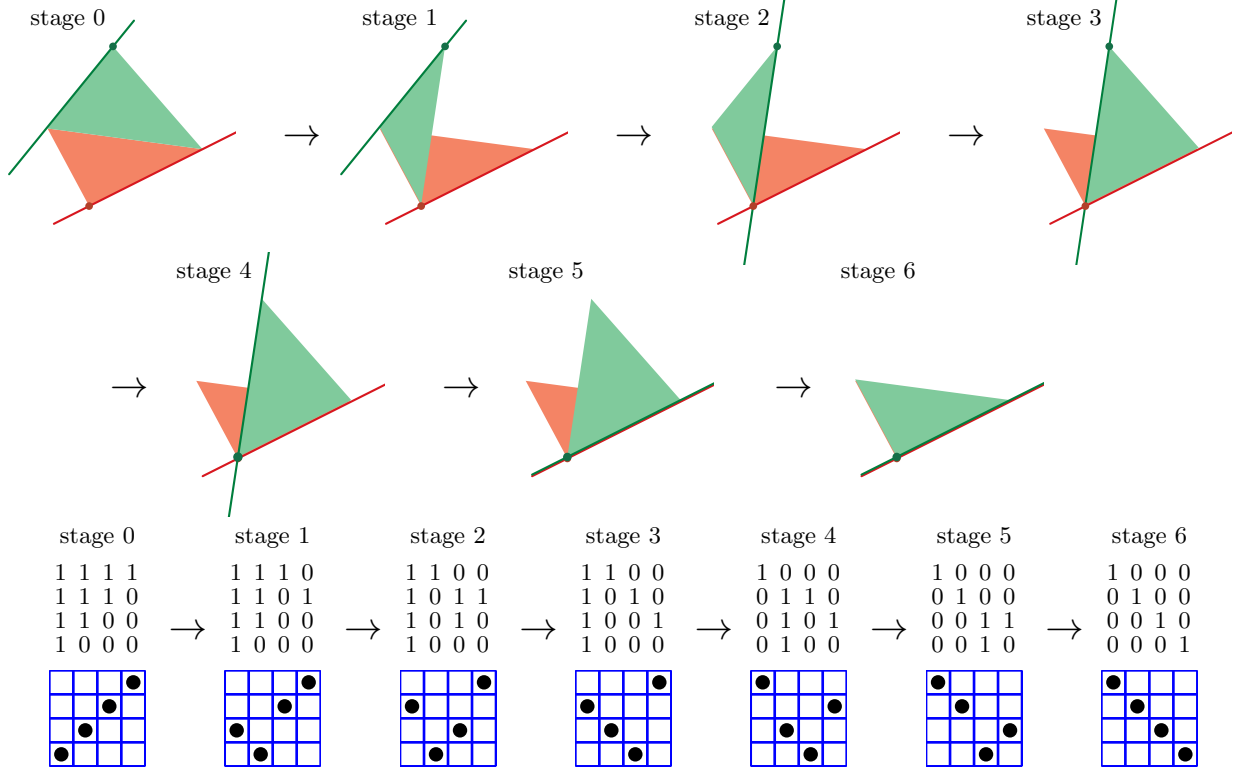


Figure 3: Specialization of the moving flag to the fixed flag.

the moving flag and checkerboards representing the relative positions of the two flags. We recognize this as the bubble sort of the black checkers.

From stage 0 to stage 1 only the moving plane moves; the line and point are fixed. The plane moves until it contains the fixed point of the fixed flag. From stage 1 to stage 2 only the moving line moves, until it too contains the fixed point. Then the moving plane moves again (to contain the fixed line); then the moving point; then the moving line; then the moving plane. At the end the two flags coincide. We describe the solution to the problem of four lines using the Littlewood-Richardson homotopy. Let $\ell_1, \ell_2, \ell_3, \ell_4$ be the four lines, where ℓ_1 is the line of the fixed flag, ℓ_2 the line of the moving flag, and ℓ_3 and ℓ_4 are two other general lines. The family of lines meeting ℓ_1 and ℓ_2 is two-dimensional; it is parameterized by $\ell_1 \times \ell_2$, as any line meeting ℓ_1 and ℓ_2 is determined by the points where it meets them. On this parameterized surface, we seek those points corresponding to lines satisfying the further condition (*) of meeting ℓ_3 and ℓ_4 . Between stages 0 and 1, nothing changes, but in moving to stage 2, ℓ_2 moves to intersect ℓ_1 . There are now two distinct two-dimensional families of lines meeting both ℓ_1 and ℓ_2 : (a) those lines lying in the plane P containing ℓ_1 and ℓ_2 and (b) those lines in space passing through the point $p = \ell_1 \cap \ell_2$. We now impose the additional condition (*) on both of these cases. In case (a), ℓ_i meets P in a point p_i ($i = 3, 4$), so there is one line in P meeting ℓ_3 and ℓ_4 , namely $\overline{p_3 p_4}$. In case (b), there is one line through p meeting ℓ_3 and ℓ_4 , namely $\overline{p, \ell_3 \cap \ell_4}$. After this, the only change is that the plane P , which equals the moving plane after stage 3, and rotates into the fixed plane between stages 5 and 6. To solve the original problem, we reverse this process, starting

with the two solutions in cases (a) and (b), and reversing the specialization.

Note that we have reduced one problem involving 4 brackets to two problems involving 3 brackets.

Figure 4 shows the geometry and algebra behind this discussion. The top shows the geometry and the bottom gives the checker description. It also shows the matrices parameterizing the two-dimensional families of lines in each case. The parameterization is explicitly described in §5.

This single example is sufficient to understand the general case. The initial position of the red checkers is as follows. The intersection of the k -plane with the moving flag M determines the rows of the red checkers, and the intersection with the fixed flag F determines their columns, and they are arranged from southwest to northeast. The movement of the moving flag in arbitrary dimension is analogous to the specific case described here, and is described by a sequence of moves of black checkers. The movement of the black checkers determines the movement of the red checkers (see §7), and at each stage, there are one or two choices. When there are two choices, the underlying geometry is essentially the same as in the example above. When there is one choice, often the underlying geometry does not change, but the parameterization changes.

4. THE LITTLEWOOD-RICHARDSON HOMOTOPY

We first explain how the geometric Littlewood-Richardson rule gives equations and homotopies for solving Schubert problems, and then illustrate that with two specific examples coming from the problem of four lines.

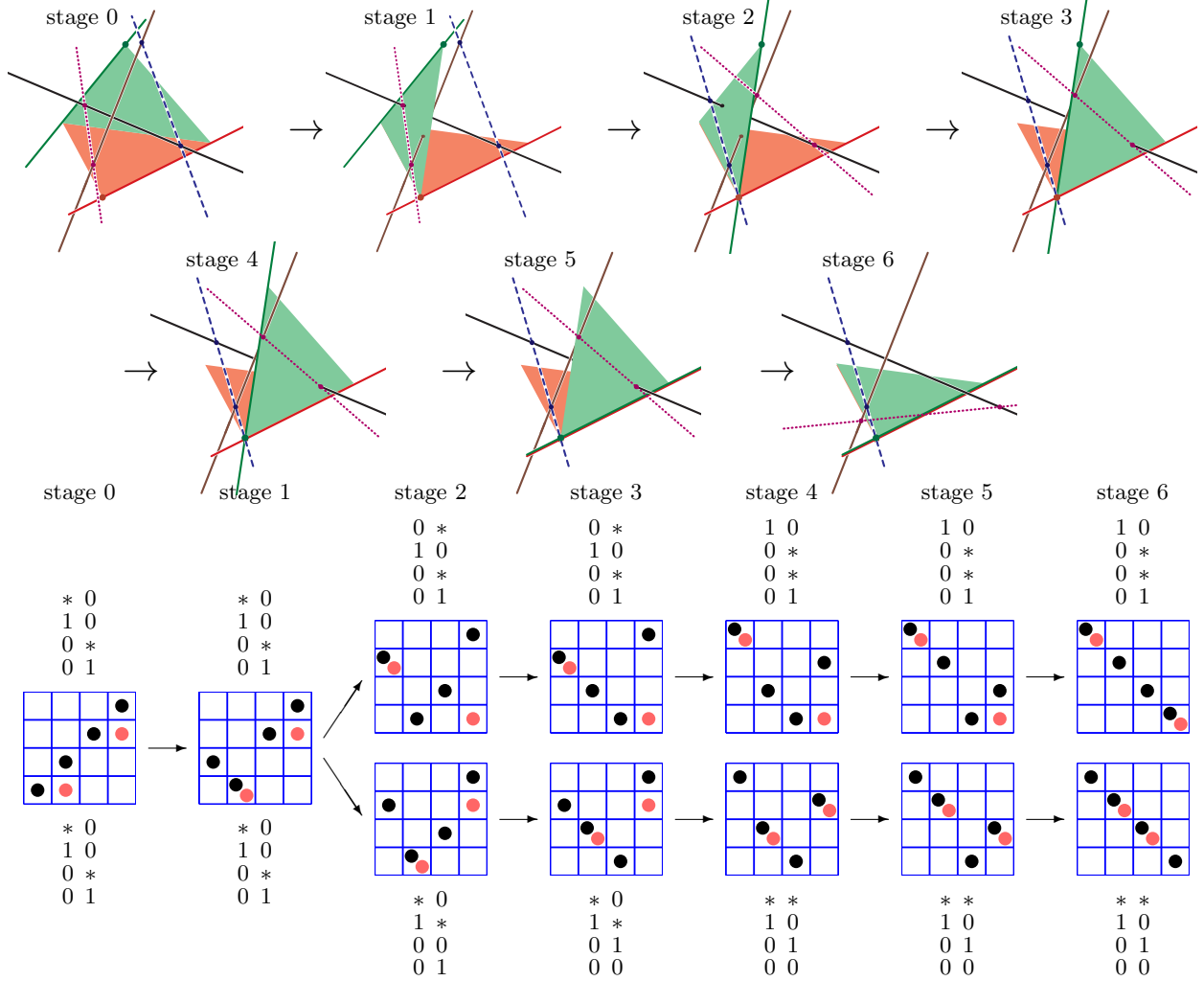


Figure 4: Resolving the problem of four lines.

In the geometric Littlewood-Richardson rule the intersection $\Omega_\omega(F) \cap \Omega_\tau(M)$ breaks into components which eventually become Schubert varieties $\Omega_\sigma(F)$ as the moving flag M specializes to coincide with the fixed flag F . At each stage, the components correspond to checkerboards.

A checkerboard encodes the relative positions of the fixed and moving flags as well as a representation X , called a localization pattern, of the general element in the corresponding component. Specifically, X is a $n \times k$ matrix whose entries are either 0, 1, or indeterminates such that the $n \times k$ matrix MX is a general point in that component. In §5 we explain how to obtain a localization pattern from its checkerboard.

Given a Schubert problem,

$$\Omega_\omega(F) \cap \Omega_\tau(M) \cap \Omega_{\rho_1}(N^1) \cap \cdots \cap \Omega_{\rho_s}(N^s), \quad (9)$$

the intersection of the last s Schubert varieties is expressed as rank conditions on (minors of) matrices $[Y|N_j^i]$ (3), where Y is a general $n \times k$ matrix representing a general k -plane. Write this system of minors succinctly as $P(Y) = 0$. When X is a localization pattern for a checkerboard in the degeneration of $\Omega_\omega(F) \cap \Omega_\tau(M)$ (as in §5), the points in the corresponding component that also lie in the last s Schubert

varieties in (9) are the solutions to the system $P(MX) = 0$.

Reversing the specialization of the flags F and M is the generalization sequence. Between adjacent stages i and $i+1$ of the generalization sequence, the moving flag is $M(t)$ for $t \in [0, 1]$. Then the homotopy connecting these stages is

$$P(M(t)X) = 0 \quad (10)$$

for $t \in [0, 1]$. When $t = 0$, we are in stage i and when $t = 1$, we are in stage $i+1$. The generalization of the moving flag is described in more detail in §6. We explain how the red checkers move in §7, and then how the localization patterns for different stages fit together.

We illustrate this with some examples from Figure 4. For $X \in \Omega_{[2 \ 4]}(F) \cap \Omega_{[2 \ 4]}(M)$, we have

$$X = \begin{bmatrix} x_{11} & 0 \\ 1 & 0 \\ 0 & x_{32} \\ 0 & 1 \end{bmatrix} \quad \begin{aligned} F &= [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4] \\ M &= [\mathbf{e}_4, \mathbf{e}_3, \mathbf{e}_2, \mathbf{e}_1] \\ \text{for any } x_{11} \text{ and } x_{32} : \\ \dim(X \cap \langle \mathbf{e}_1, \mathbf{e}_2 \rangle) &= 1, \\ \dim(X \cap \langle \mathbf{e}_4, \mathbf{e}_3 \rangle) &= 1, \\ \dim(X \cap \langle \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4 \rangle) &= 2. \end{aligned} \quad (11)$$

In the first stage of Figure 4, the plane in the moving flag

rotates about its line until it meets the fixed point. As the line in the moving flag does not move, there is no homotopy, only a change of coordinates, as illustrated in Figure 5 for a

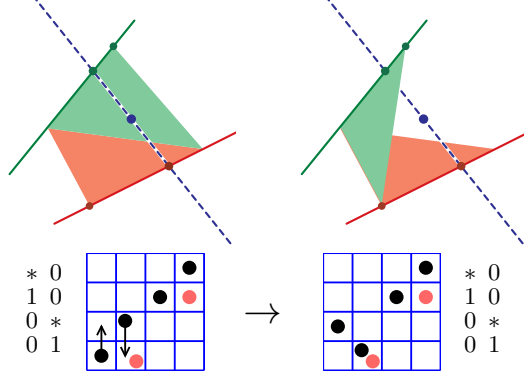


Figure 5: No homotopy, only change of coordinates.

line meeting two lines and a fixed point in three-space, and as discussed at the end of §7. The corresponding coordinate transformation is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_{11} & 0 \\ 1 & 0 \\ 0 & x_{32} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x_{11} & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & x_{32}-1 \end{bmatrix} \quad (12)$$

$$\equiv \begin{bmatrix} x_{11} & 0 \\ 1 & 0 \\ 0 & 1/(x_{32}-1) \\ 0 & 1 \end{bmatrix}. \quad (13)$$

When the red checkers swap rows, we use a homotopy, shown in Figure 6, also for the case of a line meeting two lines and a fixed point. This homotopy has coordinates

$$X(t) = \begin{bmatrix} x_{12}t & x_{12} \\ x_{32} & 0 \\ x_{32}t & x_{32} \\ 0 & 1 \end{bmatrix}. \quad (14)$$

At $t = 0$ we see that $X(0)$ fits the pattern on the right in Figure 6, while at $t = 1$ a coordinate change brings $X(1)$ into the pattern on the left. With linear combinations of the two columns we find generators for the line that fit the columns of the pattern.

5. LOCALIZATION PATTERNS

We describe coordinates for each component corresponding to a checkerboard: given two flags M and F in relative position described by the black checkers, it is the space of k -planes meeting M and F in the manner specified by the positions of the red checkers. The black checkers correspond to a basis of both F and M . Each red checker is a basis element for the k -plane and it lies in the space spanned by the black checkers weakly to its northwest.

While special cases were shown in Figure 4, we illustrate the general case with an example. In the checkerboard of Figure 7, one black checker (in row D) is descending. Red checkers are distributed along the sorted black checkers (regions B and E), as well as in the pre-sorted region (regions A , C , D , and F); in the latter region, they are distributed from

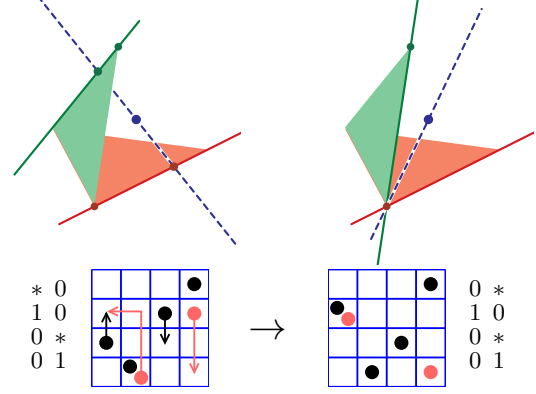


Figure 6: Homotopy, as red checkers swap rows.

the southwest to the northeast as shown. The corresponding localization pattern, which is expressed with respect to the basis of M , is shown in Figure 7.

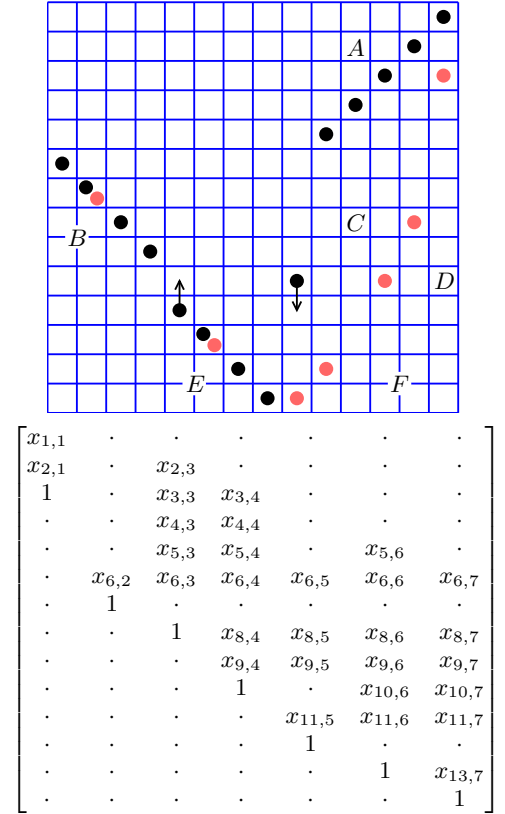


Figure 7: Coordinates corresponding to a checkerboard. Entries $\cdot$ in the coordinate matrix are 0.

We discuss the linking of localization patterns between stages after we describe the movement of checkers in §7.

6. GENERALIZING THE MOVING FLAG

Underlying the geometric Littlewood-Richardson rule is the sequence of specializations (analogous to Figure 3) in which the moving flag M successively moves to coincide with the fixed flag F . Reversing this gives the generalization sequence in which M emerges from F .

The generalization of the moving flag M is as follows. Throughout, the fixed flag is

$$F = \{\langle \mathbf{e}_1 \rangle \subset \langle \mathbf{e}_1, \mathbf{e}_2 \rangle \subset \langle \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle \cdots\}. \quad (15)$$

Initially, the moving flag M coincides with F . We let $\mathbf{m}'_i(t)$ describe the vectors during the generalization ($t = 0$ corresponds to the specialized case, and $t = 1$ corresponds to the generalized case), and $\mathbf{m}''_i$ describe the vectors after the generalization. At time t ,

$$M(t) = \{\langle \mathbf{m}'_1(t) \rangle \subset \langle \mathbf{m}'_1(t), \mathbf{m}'_2(t) \rangle \subset \cdots\}. \quad (16)$$

In the checker diagram, at each stage the black checkers in rows r and $r + 1$ swap rows, for some r . Set

$$\mathbf{m}_i = \mathbf{m}'_i(t) = \mathbf{m}''_i \quad \text{for } i \neq r, r + 1. \quad (17)$$

These different notations for the same vector keep track of whether we are talking about $t = 0$, general t , or $t = 1$.

$$\mathbf{m}_r = \mathbf{m}''_{r+1} (= \mathbf{m}'_r(0) = \mathbf{m}'_{r+1}(0)), \quad (18)$$

$$\mathbf{m}_{r+1} = \mathbf{m}''_{r+1} - \mathbf{m}''_r (= \mathbf{m}'_{r+1}(0) - \mathbf{m}'_r(0)). \quad (19)$$

$$\mathbf{m}'_r(t) = t\mathbf{m}''_r + (1-t)\mathbf{m}''_{r+1} = \mathbf{m}''_{r+1} - t\mathbf{m}_{r+1}, \quad (20)$$

$$\mathbf{m}'_i(t) = \mathbf{m}''_i \quad \text{for all other } i. \quad (21)$$

Thus $\mathbf{m}'_i(1) = \mathbf{m}''_i$ for all i .

It is convenient to describe the homotopy in terms of matrices. Here are the generalizing moves from Figure 3.

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \gamma_{31} & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (22)$$

$$\rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \gamma_{21} & 1 & 0 \\ 0 & \gamma_{31} & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} \gamma_{11} & 1 & 0 & 0 \\ \gamma_{21} & 0 & \gamma_{22} & 1 \\ \gamma_{31} & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (23)$$

$$\rightarrow \begin{bmatrix} \gamma_{11} & 1 & 0 & 0 \\ \gamma_{21} & 0 & \gamma_{22} & 1 \\ \gamma_{31} & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} \gamma_{11} & \gamma_{12} & 1 & 0 \\ \gamma_{21} & \gamma_{22} & 0 & 1 \\ \gamma_{31} & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (24)$$

$$\rightarrow \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 1 \\ \gamma_{21} & \gamma_{22} & 1 & 0 \\ \gamma_{31} & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \quad (25)$$

Here, γ_{ij} are general complex numbers. For example, the second matrix in (24) corresponds to stage 1, and we see that the moving plane, (the projectivization of) the span of the first three columns, indeed contains the fixed point, as $\mathbf{e}_1$ is in the span of those three column vectors, in agreement with Figure 3.

The arrows represent the movement of the flag M , which we parametrize using our homotopy parameter $t \in [0, 1]$. For example, the next to last deformation is

$$\begin{bmatrix} \gamma_{11} & 1 & 0 & 0 \\ \gamma_{21} & 0 & \gamma_{22} & 1 \\ \gamma_{31} & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \gamma_{12}t & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (26)$$

$$= \begin{bmatrix} \gamma_{11} & \gamma_{12}t & 1 & 0 \\ \gamma_{21} & \gamma_{22} & 0 & 1 \\ \gamma_{31} & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} =: M(t). \quad (27)$$

The gradual introduction of the random constants γ_{ij} in the moving flag is the analog here of the gamma trick [15] to ensure the regularity of the solution paths. By this gamma trick, for all t , except for a finite number of choices of γ_{ij} , the solution paths contain only regular points.

The Littlewood-Richardson homotopies operate on randomly generated complex flags. To move to flags with specific coordinates, we use coefficient-parameter [12] or cheater homotopies [10].

7. MOVEMENT OF RED CHECKERS

In the geometric Littlewood-Richardson rule, the black checkers start out on the anti-diagonal, and a bubble sort is performed which moves them to the diagonal. This is indicated in Figures 3, 4, and 7. In each of the $\binom{n}{2}$ steps, one black checker descends and another rises as in Figure 8. The

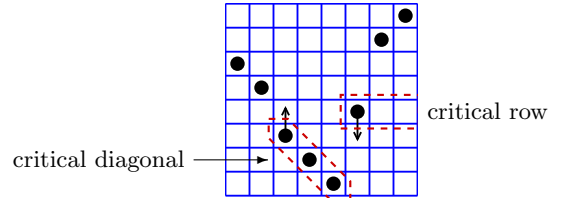


Figure 8: Critical row and critical diagonal.

descending checker is in the critical row and the ascending checker is at the top left of the critical diagonal.

To resolve the intersection $\Omega_\omega(F) \cap \Omega_\tau(M)$, we initially place red checkers as follows. The intersection of the k -plane with the moving flag M determines the rows of the red checkers, and the intersection with the fixed flag F determines their columns, and they are arranged from southwest to northeast. As the black checkers move, they induce a motion of the red checkers. There will be nine cases to consider. In eight, the motion is determined, while in the ninth case there are sometimes two choices as in Figure 4.

The cases are determined by the answers to two questions, each of which has three answers.

1. Where is the top red checker in the critical diagonal?
 - (a) In the rising checker's square.
 - (b) Elsewhere in the critical diagonal.
 - (c) There is no red checker in the critical diagonal.
2. Where is the red checker in the critical row?
 - (α) In the descending checker's square.
 - (β) Elsewhere in the critical row.
 - (γ) There is no red checker in the critical row.

Table 1 shows the movement of the checkers in these nine cases. The rows correspond to the answers to the first question and the columns to the answers of the second question. Only the relevant part of each checkerboard is shown.

In case (b, β) there are two possibilities, which can both occur—this is when a component breaks into two components in the geometric Littlewood-Richardson rule. The second of these (where the red checkers swap rows) only occurs if there are no other red checkers in the rectangle between the two, which we call blockers. Figure 9 shows a blocker.

	α	β	γ
a			
b			
c			

Table 1: Movement of red checkers.

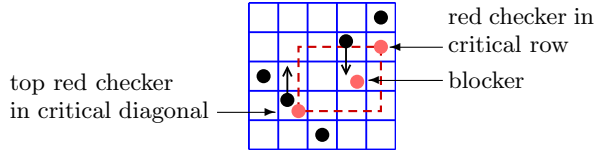


Figure 9: a blocker.

To track solutions to Schubert problems between adjacent stages in the generalization sequence, we need uniform coordinates corresponding to two adjacent diagrams—for example, two boards connected by an arrow in Figure 4. We can then track solutions from one board to the more generalized board. There are three cases to consider.

In trivial cases, such as the first arrow in Figure 4, which is case (c, γ) of Table 1, the coordinates do not change because the underlying geometry is constant.

We describe one of the nontrivial examples of the coordinates linking two stages, that of the lower arrow between stage 1 and stage 2 in Figure 4 (left case of (b, β)). We follow the vector corresponding to the red checker in the bottom row. Throughout the degeneration (as t goes from 0 to 1), we write its vector as $\mathbf{m}_4 + x\mathbf{m}'_2(t)$. As $\mathbf{m}'_2(1) = \mathbf{m}'_2$ and $\mathbf{m}'_2(0) = \mathbf{m}'_3$ (see (18)–(21) of §6 with $r = 2$), we see that the reason for the change of row of the $*$ in the matrix in Figure 4 is just a renaming of the variable.

The third case, where the two red checkers swap rows, is more subtle, and an example was described at the end of §4.

8. SOLVING SCHUBERT PROBLEMS

The global structure of the Littlewood-Richardson homotopy is encoded by a graded poset. This records the branching of Schubert varieties that occur in when running the geometric Littlewood-Richardson rule through successive specializations of their defining flags, equivalently, moving checkers as in §7.

We construct the poset for a Schubert problem

$$\Omega_{\omega^1}(F^1) \cap \Omega_{\omega^2}(F^2) \cap \cdots \cap \Omega_{\omega^s}(F^s). \quad (28)$$

First, use the geometric Littlewood-Richardson rule to resolve the first intersection

$$\Omega_{\omega^1}(F^1) \cap \Omega_{\omega^2}(F^2) \sim \sum_{\sigma} c_{\omega^1, \omega^2}^{\sigma} \Omega_{\sigma}(F^1). \quad (29)$$

The top of the poset is the bracket ω^1 , which branches to those brackets σ appearing in the sum. The edge $\omega^1 \rightarrow \sigma$ occurs with multiplicity $c_{\omega^1, \omega^2}^{\sigma}$. Geometrically, we have the disjunction of Schubert problems

$$\left(\sum_{\sigma} c_{\omega^1, \omega^2}^{\sigma} \Omega_{\sigma}(F^1) \right) \cap \Omega_{\omega^3}(F^3) \cap \cdots \cap \Omega_{\omega^s}(F^s), \quad (30)$$

and we resolve each $\Omega_{\sigma}(F^1) \cap \Omega_{\omega^3}(F^3)$ with the geometric Littlewood-Richardson rule, further building the poset, and continue in this fashion.

The penultimate stage has the form

$$\left(\sum_{\sigma} C^{\sigma} \Omega_{\sigma}(F^1) \right) \cap \Omega_{\omega^s}(F^s), \quad (31)$$

where C^{σ} are the multiplicities. This is resolved via (4), so the only term in the sum which contributes is when $\sigma^{\vee} = \omega^s$, and the final Schubert variety is $\Omega_{[1 \ 2 \ \dots \ k]}(F^1)$.

The global structure of the Littlewood-Richardson homotopy is to begin with the solution $\Omega_{[1 \ 2 \ \dots \ k]}(F^1)$ at the bottom of our poset, and continue this solution along homotopies corresponding to the edges of the poset. Each edge is a sequence of $\binom{n}{2}$ homotopies or coordinate changes corresponding to running the geometric Littlewood-Richardson rule backwards, as explained in §5, §6, and §7. In this way, we iteratively build solutions to the Schubert-type problems corresponding to the nodes of this poset.

For example, suppose that we have the Schubert problem $[2 \ 4 \ 6][2 \ 5 \ 6]^3$. This is resolved in the geometric Littlewood-Richardson rule as

$$[2 \ 4 \ 6][2 \ 5 \ 6]^3 = (1[2 \ 3 \ 5] + 1[1 \ 4 \ 5] + 1[1 \ 3 \ 6])[2 \ 5 \ 6]^2 \quad (32)$$

$$= (2[1 \ 3 \ 4] + 2[1 \ 2 \ 5])[2 \ 5 \ 6] \quad (33)$$

$$= 2[1 \ 2 \ 3]. \quad (34)$$

The poset corresponding to the Littlewood-Richardson homotopies is shown in Figure 10. The multiplicities in front

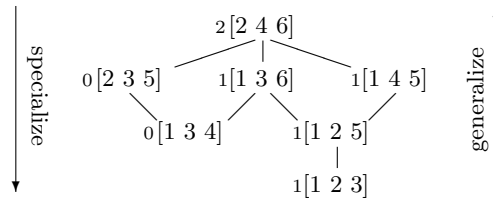


Figure 10: Poset to resolve $[2 \ 4 \ 6][2 \ 5 \ 6]^3$.

of the brackets are the number of solutions tracked to the given Schubert variety.

The number of solution paths is one of the three factors that determine the cost of the homotopies. Another factor is the complexity of the polynomials that express the intersection conditions. The current implementation performs a Laplace expansion on the minors to elaborate all conditions (1). Locally, for use during path following, an overdetermined system of p equations in q unknowns is multiplied with a q -by- p matrix of randomly generated complex coefficients to obtain square linear systems in the application of Newton's method. The third factor in the cost lies in the complex and real geometry of the solution paths. In practice it turns out that solving a generic complex instance with the Pieri homotopies is in general always faster than running a cheater homotopy using the solutions of a generic

complex instance as start solutions to solve a generic real instance. This experience also applies to solving general Schubert problems with Littlewood-Richardson homotopies.

9. COMPUTATIONAL EXPERIMENTS

Littlewood-Richardson homotopies are available in PHCpack [21] since release 2.3.46. Release 2.3.52 contains `LRhomotopies.m2`, an interface to solve Schubert problems in Macaulay 2 [5]. Via `phc -e option #4` we resolve intersection conditions and Littlewood-Richardson homotopies are available via option #5.

Below we list sample timings for solving some small Schubert problems on one core of a Mac OS X 2.2 Ghz:

- $[2\ 4]^4 = 2$ takes 5 milliseconds,
- $[2\ 4\ 6]^3 = 2$ takes 169 milliseconds,
- $[2\ 5\ 8]^2[4\ 6\ 8] = 2$ takes 2.556 seconds,
- $[2\ 4\ 6\ 8]^2[2\ 5\ 7\ 8] = 3$ takes 8.595 seconds.

10. REFERENCES

- [1] R.W. Brockett and C.I. Byrnes. Multivariate Nyquist criteria, root loci, and pole placement: a geometric viewpoint. *IEEE Trans. Automat. Control*, 26(1):271–284, 1981.
- [2] C.I. Byrnes. Pole assignment by output feedback. In H. Nijmayer and J.M. Schumacher, editors, *Three Decades of Mathematical Systems Theory*, volume 135 of *Lecture Notes in Control and Inform. Sci.*, pages 13–78. Springer-Verlag, 1989.
- [3] A. Eremenko and A. Gabrielov. Pole placement by static output feedback for generic linear systems. *SIAM J. Control Optim.*, 41(1):303–312, 2002.
- [4] W. Fulton. *Young Tableau. With Applications to Representation Theory and Geometry*. Cambridge University Press, 1997.
- [5] D.R. Grayson and M.E. Stillman. Macaulay 2, a software system for research in algebraic geometry. Available at <http://www.math.uiuc.edu/Macaulay2/>.
- [6] C. Hillar, L. Garcia-Puente, A. Martin del Campo, J. Ruffo, Z. Teitler, S.L. Johnson, and F. Sottile. Experimentation at the frontiers of reality in Schubert calculus. *Contemporary Mathematics*, to appear. Preprint [arXiv:0906.2497v2](https://arxiv.org/abs/0906.2497v2) [math.AG].
- [7] B. Huber, F. Sottile, and B. Sturmfels. Numerical Schubert calculus. *J. Symbolic Computation*, 26(6):767–788, 1998.
- [8] B. Huber and J. Verschelde. Pieri homotopies for problems in enumerative geometry applied to pole placement in linear systems control. *SIAM J. Control Optim.*, 38(4):1265–1287, 2000.
- [9] A. Leykin and F. Sottile. Galois group of Schubert problems via homotopy continuation. *Math. Comp.*, 78(267):1749–1765, 2009.
- [10] T.Y. Li, T. Sauer, and J.A. Yorke. The cheater’s homotopy: an efficient procedure for solving systems of polynomial equations. *SIAM J. Numer. Anal.*, 26(5):1241–1251, 1989.
- [11] T.Y. Li, X. Wang, and M. Wu. Numerical Schubert calculus by the Pieri homotopy algorithm. *SIAM J. Numer. Anal.*, 20(2):578–600, 2002.
- [12] A.P. Morgan and A.J. Sommese. Coefficient-parameter polynomial continuation. *Appl. Math. Comput.*, 29(2):123–160, 1989. Errata: *Appl. Math. Comput.* 51:207(1992).
- [13] M.S. Ravi, J. Rosenthal, and X. Wang. Dynamic pole placement assignment and Schubert calculus. *SIAM J. Control and Optimization*, 34(3):813–832, 1996.
- [14] H. Schubert. Anzahl-Bestimmungen für lineare Räume beliebiger Dimension. *Acta. Math.*, 8:97–118, 1886.
- [15] A.J. Sommese and C.W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific, 2005.
- [16] F. Sottile. Pieri’s formula via explicit rational equivalence. *Canad. J. Math.*, 49(6):1281–1298, 1997.
- [17] F. Sottile. Real Schubert calculus: polynomial systems and a conjecture of Shapiro and Shapiro. *Experiment. Math.*, 9(2):161–182, 2000.
- [18] F. Sottile. Enumerative real algebraic geometry. In S. Basu and L. Gonzalez-Vega, editors, *Algorithmic and Quantitative Real Algebraic Geometry*, pages 139–180. AMS, 2003.
- [19] F. Sottile. Frontiers of reality in Schubert calculus. *Bulletin of the AMS*, 47(1):31–71, 2010.
- [20] R. Vakil. A geometric Littlewood-Richardson rule. *Annals of Math.*, 164(2):376–421, 2006.
- [21] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, 1999. Software available at <http://www.math.uic.edu/~jan>.
- [22] J. Verschelde. Numerical evidence for a conjecture in real algebraic geometry. *Experimental Mathematics*, 9(2):183–196, 2000.
- [23] J. Verschelde and Y. Wang. Computing dynamic output feedback laws. *IEEE Trans. Automat. Control*, 49(8):1393–1397, 2004.
- [24] J. Verschelde and Y. Wang. Computing feedback laws for linear systems with a parallel Pieri homotopy. In Y. Yang, editor, *Proceedings of the 2004 International Conference on Parallel Processing Workshops, 15-18 August 2004, Montreal, Quebec, Canada. High Performance Scientific and Engineering Computing*, pages 222–229. IEEE Computer Society, 2004.

Triangular Decomposition of Semi-Algebraic Systems

Changbo Chen
University of Western Ontario
cchen252@csd.uwo.ca

Marc Moreno Maza
University of Western Ontario
moreno@csd.uwo.ca

James H. Davenport
University of Bath
J.H.Davenport@bath.ac.uk

Bican Xia
Peking University
xbc@math.pku.edu.cn

John P. May
Maplesoft
jmay@maplesoft.com

Rong Xiao
University of Western Ontario
rong@csd.uwo.ca

ABSTRACT

Regular chains and triangular decompositions are fundamental and well-developed tools for describing the complex solutions of polynomial systems. This paper proposes adaptations of these tools focusing on solutions of the real analogue: semi-algebraic systems.

We show that any such system can be decomposed into finitely many *regular semi-algebraic systems*. We propose two specifications of such a decomposition and present corresponding algorithms. Under some assumptions, one type of decomposition can be computed in singly exponential time w.r.t. the number of variables. We implement our algorithms and the experimental results illustrate their effectiveness.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—*Algebraic algorithms, Analysis of algorithms*

General Terms

Algorithms, Experimentation, Theory

Keywords

regular semi-algebraic system, regular chain, triangular decomposition, border polynomial, fingerprint polynomial set

1. INTRODUCTION

Regular chains, the output of triangular decompositions of systems of polynomial equations, enjoy remarkable properties. Size estimates play in their favor [12] and permit the design of modular [13] and fast [17] methods for computing triangular decompositions. These features stimulate the development of algorithms and software for solving polynomial systems via triangular decompositions.

For the fundamental case of semi-algebraic systems with rational number coefficients, to which this paper is devoted, we observe that several algorithms for studying the real solutions of such systems take advantage of the structure of a

regular chain. Some are specialized to isolating the real solutions of systems with finitely many complex solutions [23, 10, 3]. Other algorithms deal with parametric polynomial systems via real root classification (RRC) [25] or with arbitrary systems via cylindrical algebraic decompositions (CAD) [9].

In this paper, we introduce the notion of a *regular semi-algebraic system*, which in broad terms is the “real” counterpart of the notion of a regular chain. Then we define two notions of a *decomposition of a semi-algebraic system*: one that we call *lazy triangular decomposition*, where the analysis of components of strictly smaller dimension is deferred, and one that we call *full triangular decomposition* where all cases are worked out. These decompositions are obtained by combining triangular decompositions of algebraic sets over the complex field with a special Quantifier Elimination (QE) method based on RRC techniques.

Regular semi-algebraic system. Let T be a regular chain of $\mathbb{Q}[x_1, \dots, x_n]$ for some ordering of the variables $\mathbf{x} = x_1, \dots, x_n$. Let $\mathbf{u} = u_1, \dots, u_d$ and $\mathbf{y} = y_1, \dots, y_{n-d}$ designate respectively the variables of $\mathbf{x}$ that are free and algebraic w.r.t. T . Let $P \subset \mathbb{Q}[\mathbf{x}]$ be finite such that each polynomial in P is regular w.r.t. the saturated ideal of T . Define $P_{>} := \{p > 0 \mid p \in P\}$. Let Q be a quantifier-free formula of $\mathbb{Q}[\mathbf{x}]$ involving only the variables of $\mathbf{u}$. We say that $R := [Q, T, P_{>}]$ is a *regular semi-algebraic system* if:

- (i) Q defines a non-empty open semi-algebraic set S in $\mathbb{R}^d$,
- (ii) the regular system $[T, P]$ specializes well at every point u of S (see Section 2 for this notion),
- (iii) at each point u of S , the specialized system $[T(u), P(u)_{>}]$ has at least one real zero.

The zero set of R , denoted by $Z_{\mathbb{R}}(R)$, is defined as the set of points $(u, y) \in \mathbb{R}^d \times \mathbb{R}^{n-d}$ such that $Q(u)$ is true and $t(u, y) = 0$, $p(u, y) > 0$, for all $t \in T$ and all $p \in P$.

Triangular decomposition of a semi-algebraic system. In Section 3 we show that the zero set of any semi-algebraic system $\mathfrak{S}$ can be decomposed as a finite union (possibly empty) of zero sets of regular semi-algebraic systems. We call such a decomposition a *full triangular decomposition* (or simply *triangular decomposition* when clear from context) of $\mathfrak{S}$, and denote by `RealTriangularize` an algorithm to compute it. The proof of our statement relies on triangular decompositions in the sense of Lazard (see Section 2 for this notion) for which it is not known whether or not they can be computed in singly exponential time w.r.t. the number of variables. Meanwhile, we are hoping to obtain an algorithm for decomposing semi-algebraic systems (certainly under some genericity assumptions) that would fit in that complexity class. Moreover, we observe that, in practice, full triangular

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

decompositions are not always necessary and that providing information about the components of maximum dimension is often sufficient. These theoretical and practical motivations lead us to a weaker notion of a decomposition of a semi-algebraic system.

Lazy triangular decomposition of a semi-algebraic system. Let $\mathfrak{S} = [F, N_{\geq}, P_{>}, H_{\neq}]$ (see Section 3 for this notation) be a semi-algebraic system of $\mathbb{Q}[\mathbf{x}]$ and $Z_{\mathbb{R}}(\mathfrak{S}) \subseteq \mathbb{R}^n$ be its zero set. Denote by d the dimension of the constructible set $\{x \in \mathbb{C}^n \mid f(x) = 0, g(x) \neq 0, \text{ for all } f \in F, g \in P \cup H\}$. A finite set of regular semi-algebraic systems $R_i, i = 1 \dots t$ is called a *lazy triangular decomposition* of $\mathfrak{S}$ if

- $\cup_{i=1}^t Z_{\mathbb{R}}(R_i) \subseteq Z_{\mathbb{R}}(\mathfrak{S})$ holds, and
- there exists $G \subset \mathbb{Q}[\mathbf{x}]$ such that the real-zero set $Z_{\mathbb{R}}(G) \subset \mathbb{R}^n$ contains $Z_{\mathbb{R}}(\mathfrak{S}) \setminus (\cup_{i=1}^t Z_{\mathbb{R}}(R_i))$ and the complex-zero set $V(G) \subset \mathbb{C}^n$ either is empty or has dimension less than d .

We denote by `LazyRealTriangularize` an algorithm computing such a decomposition. In the implementation presented hereafter, `LazyRealTriangularize` outputs additional information in order to continue the computations and obtain a full triangular decomposition, if needed. This additional information appears in the form of *unevaluated function calls*, explaining the usage of the adjective *lazy* in this type of decompositions.

Complexity results for lazy triangular decomposition. In Section 4, we provide a running time estimate for computing a lazy triangular decomposition of the semi-algebraic system $\mathfrak{S}$ when $\mathfrak{S}$ has no inequations nor inequalities, (that is, when $N_{\geq} = P_{>} = H_{\neq} = \emptyset$ holds) and when F generates a strongly equidimensional ideal of dimension d . We show that one can compute such a decomposition in time singly exponential w.r.t. n . Our estimates are not sharp and are just meant to reach a singly exponential bound. We rely on the work of J. Renegar [20] for QE. In Sections 5 and 6 we turn our attention to algorithms that are more suitable for implementation even though they rely on sub-algorithms with a doubly exponential running time w.r.t. d .

A special case of quantifier elimination. By means of triangular decomposition of algebraic sets over $\mathbb{C}$, triangular decomposition of semi-algebraic systems (both full and lazy) reduces to a special case of QE. In Section 5, we implement this latter step via the concept of a *fingerprint polynomial set*, which is inspired by that of a *discrimination polynomial set* used for RRC in [25, 24].

Implementation and experimental results. In Section 6 we describe the algorithms that we have implemented for computing triangular decompositions (both full and lazy) of semi-algebraic systems. Our MAPLE code is written on top of the `RegularChains` library. We provide experimental data for two groups of well-known problems. In the first group, each input semi-algebraic system consists of equations only while the second group is a collection of QE problems. To illustrate the difficulty of our test problems, and only for this purpose, we provide timings obtained with other well-known polynomial system solvers which are based on algorithms whose running time estimates are comparable to ours. For this first group we use the MAPLE command `Groebner:-Basis` for computing lexicographical Gröbner bases. For the second group we use a general purpose QE software: QPCAD B (in its non-interactive mode) [5]. Our experimental results show that our `LazyRealTriangularize` code can solve most of

our test problems and that it can solve more problems than the package it is compared to.

We conclude this introduction by computing a triangular decomposition of a particular semi-algebraic system taken from [6]. Consider the following question: when does $p(z) = z^3 + az + b$ have a non-real root $x + iy$ satisfying $xy < 1$? This problem can be expressed as $(\exists x)(\exists y)[f = g = 0 \wedge y \neq 0 \wedge xy - 1 < 0]$, where $f = \text{Re}(p(x + iy)) = x^3 - 3xy^2 + ax + b$ and $g = \text{Im}(p(x + iy))/y = 3x^2 - y^2 + a$.

We call our `LazyRealTriangularize` command on the semi-algebraic system $f = 0, g = 0, y \neq 0, xy - 1 < 0$ with the variable order $y > x > b > a$. Its first step is to call the `Triangularize` command of the `RegularChains` library on the algebraic system $f = g = 0$. We obtain one squarefree regular chain $T = [t_1, t_2]$, where $t_1 = g$ and $t_2 = 8x^3 + 2ax - b$, satisfying $V(f, g) = V(T)$. The second step of `LazyRealTriangularize` is to check whether the polynomials defining inequalities and inequations are regular w.r.t. the saturated ideal of T , which is the case here. The third step is to compute the so called *border polynomial set* (see Section 2) which is $B = [h_1, h_2]$ with $h_1 = 4a^3 + 27b^2$ and $h_2 = -4a^3b^2 - 27b^4 + 16a^4 + 512a^2 + 4096$. One can check that the regular system $[T, \{y, xy - 1\}]$ specializes well outside of the hypersurface $h_1h_2 = 0$. The fourth step is to compute the fingerprint polynomial set which yields the quantifier-free formula $Q = h_1 > 0$ telling us that $[Q, T, 1 - xy > 0]$ is a regular semi-algebraic system. After performing these four steps, (based on Algorithm 5, Section 6) the function call

`LazyRealTriangularize([f, g, y ≠ 0, xy - 1 < 0], [y, x, b, a])`

in our implementation returns the following:

$$\left\{ \begin{array}{ll} [\{t_1 = 0, t_2 = 0, 1 - xy > 0, h_1 > 0\}] & h_1h_2 \neq 0 \\ \% \text{LazyRealTriangularize}([t_1 = 0, t_2 = 0, f = 0, \\ h_1 = 0, 1 - xy > 0, y \neq 0], [y, x, b, a]) & h_1 = 0 \\ \% \text{LazyRealTriangularize}([t_1 = 0, t_2 = 0, f = 0, \\ h_2 = 0, 1 - xy > 0, y \neq 0], [y, x, b, a]) & h_2 = 0 \end{array} \right.$$

The above output shows that $\{[Q, T, 1 - xy > 0]\}$ forms a lazy triangular decomposition of the input semi-algebraic system. Moreover, together with the output of the recursive calls, one obtains a full triangular decomposition. Note that the cases of the two recursive calls correspond to $h_1 = 0$ and $h_2 = 0$. Since our `LazyRealTriangularize` uses the MAPLE piecewise structure for formatting its output, one simply needs to evaluate the recursive calls with the `value` command, yielding the same result as directly calling `RealTriangularize`

$$\left\{ \begin{array}{ll} [\{t_1 = 0, t_2 = 0, 1 - xy > 0, h_1 > 0\}] & h_1h_2 \neq 0 \\ [] & h_1 = 0 \\ \left\{ \begin{array}{ll} ([\{t_3 = 0, t_4 = 0, \\ h_2 = 0, 1 - xy > 0\}]) & h_3 \neq 0 \end{array} \right. & h_2 = 0 \\ [] & h_3 = 0 \end{array} \right.$$

where $t_3 = xy + 1$, $t_4 = 2a^3x - a^2b + 32ax - 48b + 18xb^2$, $h_3 = (a^2 + 48)(a^2 + 16)(a^2 + 12)$.

From this output, after some simplification, one could obtain the equivalent quantifier-free formula, $4a^3 + 27b^2 > 0$, of the original QE problem.

2. TRIANGULAR DECOMPOSITION OF ALGEBRAIC SETS

We review in the section the basic notions related to regular chains and triangular decompositions of algebraic sets. Throughout this paper, let $\mathbf{k}$ be a field of characteristic 0 and $\mathbf{K}$ be its algebraic closure. Let $\mathbf{k}[\mathbf{x}]$ be the polynomial ring over $\mathbf{k}$ and with ordered variables $\mathbf{x} = x_1 < \dots < x_n$. Let $p, q \in \mathbf{k}[\mathbf{x}]$ be polynomials. Assume that $p \notin \mathbf{k}$. Then denote by $\text{mvar}(p)$, $\text{init}(p)$, and $\text{mdeg}(p)$ respectively the greatest variable appearing in p (called the *main variable* of p), the leading coefficient of p w.r.t. $\text{mvar}(p)$ (called the *initial* of p), and the degree of p w.r.t. $\text{mvar}(p)$ (called the *main degree* of p); denote by $\text{der}(p)$ the derivative of p w.r.t. $\text{mvar}(p)$; denote by $\text{discrim}(p)$ the discriminant of p w.r.t. $\text{mvar}(p)$.

Triangular sets. Let $T \subset \mathbf{k}[\mathbf{x}]$ be a *triangular set*, that is, a set of non-constant polynomials with pairwise distinct main variables. Denote by $\text{mvar}(T)$ the set of main variables of the polynomials in T . A variable v in $\mathbf{x}$ is called *algebraic* w.r.t. T if $v \in \text{mvar}(T)$, otherwise it is said *free* w.r.t. T . If no confusion is possible, we shall always denote by $\mathbf{u} = u_1, \dots, u_d$ and $\mathbf{y} = y_1, \dots, y_m$ respectively the free and the main variables of T . Let h_T be the product of the initials of the polynomials in T . We denote by $\text{sat}(T)$ the *saturated ideal* of T : if T is the empty triangular set, then $\text{sat}(T)$ is defined as the trivial ideal $\langle 0 \rangle$, otherwise it is the ideal $\langle T \rangle : h_T^\infty$. The *quasi-component* $W(T)$ of T is defined as $V(T) \setminus V(h_T)$. Denote $\bar{W}(T) = V(\text{sat}(T))$ as the Zariski closure of $W(T)$.

Iterated resultant. Let p and q be two polynomials of $\mathbf{k}[\mathbf{x}]$. Assume q is non-constant and let $v = \text{mvar}(q)$. We define $\text{res}(p, q, v)$ as follows: if v does not appear in p , then $\text{res}(p, q, v) := p$; otherwise $\text{res}(p, q, v)$ is the resultant of p and q w.r.t. v . Let T be a triangular set of $\mathbf{k}[\mathbf{x}]$. We define $\text{res}(p, T)$ by induction: if T is empty, then $\text{res}(p, T) = p$; otherwise let v be the greatest variable appearing in T , then $\text{res}(p, T) = \text{res}(\text{res}(p, T_v, v), T_{<v})$, where T_v and $T_{<v}$ denote respectively the polynomials of T with main variables equal to and less than v .

Regular chain. A triangular set $T \subset \mathbf{k}[\mathbf{x}]$ is a *regular chain* if: either T is empty; or (letting t be the polynomial in T with maximum main variable), $T \setminus \{t\}$ is a regular chain, and the initial of t is regular w.r.t. $\text{sat}(T \setminus \{t\})$. The empty regular chain is denoted by $\emptyset$. Let $H \subset \mathbf{k}[\mathbf{x}]$. The pair $[T, H]$ is a *regular system* if each polynomial in H is regular modulo $\text{sat}(T)$. A regular chain T or a regular system $[T, H]$, is *squarefree* if for all $t \in T$, the $\text{der}(t)$ is regular w.r.t. $\text{sat}(T)$.

Triangular decomposition. Let $F \subset \mathbf{k}[\mathbf{x}]$. Regular chains $T_1, \dots, T_e$ of $\mathbf{k}[\mathbf{x}]$ form a *triangular decomposition* of $V(F)$ if either: $V(F) = \bigcup_{i=1}^e \bar{W}(T_i)$ (Kalkbrener's sense) or $V(F) = \bigcup_{i=1}^e W(T_i)$ (Lazard's sense). In this paper, we denote by **Triangularize** an algorithm, such as the one of [18], computing a triangular decomposition of the former kind.

Regularization. Let $p \in \mathbf{k}[\mathbf{x}]$. Let T be a regular chain of $\mathbf{k}[\mathbf{x}]$. Denote by **Regularize**(p, T) an operation which computes a set of regular chains $\{T_1, \dots, T_e\}$ such that (1) for each $i, 1 \leq i \leq e$, either $p \in \text{sat}(T_i)$ or p is regular w.r.t. $\text{sat}(T_i)$; (2) we have $\bar{W}(T) = \bar{W}(T_1) \cup \dots \cup \bar{W}(T_e)$, $\text{mvar}(T) = \text{mvar}(T_i)$ and $\text{sat}(T) \subseteq \text{sat}(T_i)$ for $1 \leq i \leq e$.

Good specialization [8]. Consider a squarefree regular system $[T, H]$ of $\mathbf{k}[\mathbf{u}, \mathbf{y}]$. Recall that $\mathbf{y}$ and $\mathbf{u} = u_1, \dots, u_d$ stand respectively for $\text{mvar}(T)$ and $\mathbf{x} \setminus \mathbf{y}$. Let $z = (z_1, \dots, z_d)$ be

a point of $\mathbf{K}^d$. We say that $[T, H]$ *specializes well* at z if: (i) none of the initials of the polynomials in T vanishes modulo the ideal $\langle z_1 - u_1, \dots, z_d - u_d \rangle$; (ii) the image of $[T, H]$ modulo $\langle z_1 - u_1, \dots, z_d - u_d \rangle$ is a squarefree regular system.

Border polynomial [25]. Let $[T, H]$ be a squarefree regular system of $\mathbf{k}[\mathbf{u}, \mathbf{y}]$. Let bp be the primitive and square free part of the product of all $\text{res}(\text{der}(t), T)$ and all $\text{res}(h, T)$ for $h \in H$ and $t \in T$. We call bp the *border polynomial* of $[T, H]$ and denote by **BorderPolynomial**(T, H) an algorithm to compute it. We call the set of irreducible factors of bp the *border polynomial set* of $[T, H]$. Denote by **BorderPolynomialSet**(T, H) an algorithm to compute it. Proposition 1 follows from the specialization property of subresultants and states a fundamental property of border polynomials.

PROPOSITION 1. *The system $[T, H]$ specializes well at $u \in \mathbf{K}^d$ if and only if the border polynomial $bp(u) \neq 0$.*

3. TRIANGULAR DECOMPOSITION OF SEMI-ALGEBRAIC SYSTEMS

In this section, we prove that any semi-algebraic system can be decomposed into finitely many regular semi-algebraic systems. This latter notion was defined in the introduction.

Semi-algebraic system. Consider four finite polynomial subsets $F = \{f_1, f_2, \dots, f_s\}$, $N = \{n_1, n_2, \dots, n_t\}$, $P = \{p_1, p_2, \dots, p_r\}$, and $H = \{h_1, h_2, \dots, h_\ell\}$ of $\mathbb{Q}[x_1, \dots, x_n]$. Let $N_{\geq}$ denote the set of non-negative inequalities $\{n_1 \geq 0, \dots, n_t \geq 0\}$. Let $P_{>}$ denote the set of positive inequalities $\{p_1 > 0, \dots, p_r > 0\}$. Let $H_{\neq}$ denote the set of inequations $\{h_1 \neq 0, \dots, h_\ell \neq 0\}$. We will denote by $[F, P_{>}]$ the *basic semi-algebraic system* $\{f_1 = 0, \dots, f_s = 0, p_1 > 0, \dots, p_r > 0\}$. We denote by $\mathfrak{S} = [F, N_{\geq}, P_{>}, H_{\neq}]$ the semi-algebraic system (SAS) which is the conjunction of the following conditions: $f_1 = 0, \dots, f_s = 0$, $n_1 \geq 0, \dots, n_t \geq 0$, $p_1 > 0, \dots, p_r > 0$ and $h_1 \neq 0, \dots, h_\ell \neq 0$.

Notations on zero sets. In this paper, we use “ Z ” to denote the zero set of a polynomial system, involving equations and inequations, in $\mathbb{C}^n$ and “ $Z_{\mathbb{R}}$ ” to denote the zero set of a semi-algebraic system in $\mathbb{R}^n$.

Pre-regular semi-algebraic system. Let $[T, P]$ be a square-free regular system of $\mathbb{Q}[\mathbf{u}, \mathbf{y}]$. Let bp be the border polynomial of $[T, P]$. Let $B \subset \mathbb{Q}[\mathbf{u}]$ be a polynomial set such that bp divides the product of polynomials in B . We call the triple $[B_{\neq}, T, P_{>}]$ a *pre-regular semi-algebraic system* of $\mathbb{Q}[\mathbf{x}]$. Its zero set, written as $Z_{\mathbb{R}}(B_{\neq}, T, P_{>})$, is the set $(u, y) \in \mathbb{R}^n$ such that $b(u) \neq 0$, $t(u, y) = 0$, $p(u, y) > 0$, for all $b \in B$, $t \in T$, $p \in P$. Lemma 1 and Lemma 2 are fundamental properties of pre-regular semi-algebraic systems.

LEMMA 1. *Let $\mathfrak{S}$ be a semi-algebraic system of $\mathbb{Q}[\mathbf{x}]$. Then there exists finitely many pre-regular semi-algebraic systems $[B_{i\neq}, T_i, P_{i>}]$, $i = 1 \dots e$, s.t. $Z_{\mathbb{R}}(\mathfrak{S}) = \bigcup_{i=1}^e Z_{\mathbb{R}}(B_{i\neq}, T_i, P_{i>})$.*

PROOF. The semi-algebraic system $\mathfrak{S}$ decomposes into basic semi-algebraic systems, by rewriting inequality of type $n \geq 0$ as: $n > 0 \vee n = 0$. Let $[F, P_{>}]$ be one of those basic semi-algebraic systems. If F is empty, then the triple $[P, \emptyset, P_{>}]$ is a pre-regular semi-algebraic system. If F is not empty, by Proposition 1 and the specifications of **Triangularize** and **Regularize**, one can compute finitely many squarefree regular systems $[T_i, H]$ such that $V(F) \cap Z(P_{\neq}) = \bigcup_{i=1}^e (V(T_i) \cap Z(B_{i\neq}))$ holds and where B_i is the border polynomial set of the regular system $[T_i, H]$. Hence, we have

$Z_{\mathbb{R}}(F, P_{>}) = \cup_{i=1}^e Z_{\mathbb{R}}(B_{i\neq}, T_i, P_{>})$, where each $[B_{i\neq}, T_i, P_{>}]$ is a pre-regular semi-algebraic system. $\square$

LEMMA 2. Let $[B_{\neq}, T, P_{>}]$ be a pre-regular semi-algebraic system of $\mathbb{Q}[\mathbf{u}, \mathbf{y}]$. Let h be the product of polynomials in B . The complement of the hypersurface $h = 0$ in $\mathbb{R}^d$ consists of finitely many open cells of dimension d . Let C be one of them. Then for all $\alpha \in C$, the number of real zeros of $[T(\alpha), P_{>}(\alpha)]$ is the same.

PROOF. From Proposition 1 and recursive use of Theorem 1 in [11] on the delineability of a polynomial. $\square$

LEMMA 3. Let $[B_{\neq}, T, P_{>}]$ be a pre-regular semi-algebraic system of $\mathbb{Q}[\mathbf{u}, \mathbf{y}]$. One can decide whether its zero set is empty or not. If it is not empty, then one can compute a regular semi-algebraic system $[Q, T, P_{>}]$ whose zero set in $\mathbb{R}^n$ is the same as that of $[B_{\neq}, T, P_{>}]$.

PROOF. If $T = \emptyset$, we can always test whether the zero set of $[B_{\neq}, P_{>}]$ is empty or not, for instance using CAD. If it is empty, we are done. Otherwise, defining $Q = B_{\neq} \wedge P_{>}$, the triple $[Q, T, P_{>}]$ is a regular semi-algebraic system. If T is not empty, we solve the quantifier elimination problem $\exists \mathbf{y}(B(\mathbf{u}) \neq 0, T(\mathbf{u}, \mathbf{y}) = 0, P(\mathbf{u}, \mathbf{y}) > 0)$ and let Q be the resulting formula. If Q is false, we are done. Otherwise, by Lemma 2, above each connected component of $B(\mathbf{u}) \neq 0$, the number of real zeros of the system $[B_{\neq}, T, P_{>}]$ is constant. Then, the zero set defined by Q is the union of the connected components of $B(\mathbf{u}) \neq 0$ above which $[B_{\neq}, T, P_{>}]$ possesses at least one solution. Thus, Q defines a nonempty open set of $\mathbb{R}^d$ and $[Q, T, P_{>}]$ is a regular semi-algebraic system. $\square$

THEOREM 1. Let $\mathfrak{S}$ be a semi-algebraic system of $\mathbb{Q}[\mathbf{x}]$. Then one can compute a (full) triangular decomposition of $\mathfrak{S}$, that is, as defined in the introduction, finitely many regular semi-algebraic systems such that the union of their zero sets is the zero set of $\mathfrak{S}$.

PROOF. It follows from Lemma 1 and 3. $\square$

4. COMPLEXITY RESULTS

We start this section by stating complexity estimates for basic operations on multivariate polynomials.

Complexity of basic polynomial operations. Let $p, q \in \mathbb{Q}[\mathbf{x}]$ be polynomials with respective total degrees δ_p, δ_q , and let $x \in \mathbf{x}$. Let h_p, h_q, h_{pq} and h_r be the height (that is, the bit size of the maximum absolute value of the numerator or denominator of a coefficient) of p, q , the product pq and the resultant $\text{res}(p, q, x)$, respectively. In [14], it is proved that $\text{gcd}(p, q)$ can be computed within $O(n^{2\delta+1}h^3)$ bit operations where $\delta = \max(\delta_p, \delta_q)$ and $h = \max(h_p, h_q)$. It is easy to establish that h_{pq} and h_r are respectively upper bounded by $h_p + h_q + n \log(\min(\delta_p, \delta_q) + 1)$ and $\delta_q h_p + \delta_p h_q + n \delta_q \log(\delta_p + 1) + n \delta_p \log(\delta_q + 1) + \log((\delta_p + \delta_q)!).$ Finally, let M be a $k \times k$ matrix over $\mathbb{Q}[\mathbf{x}]$. Let δ (resp. h) be the maximum total degree (resp. height) of a polynomial coefficient of M . Then $\det(M)$ can be computed within $O(k^{2n+5}(\delta + 1)^{2n}h^2)$ bit operations, see [15].

We turn now to the main subject of this section, that is, complexity estimates for a lazy triangular decomposition of a polynomial system under some genericity assumptions. Let $F \subset \mathbb{Q}[\mathbf{x}]$. A lazy triangular decomposition (as defined in the Introduction) of the semi-algebraic system $\mathfrak{S} = [F, \emptyset, \emptyset, \emptyset]$,

Algorithm 1: LazyRealTriangularize($\mathfrak{S}$)

Input: a semi-algebraic system $\mathfrak{S} = [F, \emptyset, \emptyset, \emptyset]$

Output: a lazy triangular decomposition of $\mathfrak{S}$

```

1  $\mathfrak{T} := \text{Triangularize}(F)$ 
2 for  $T_i \in \mathfrak{T}$  do
3    $bp_i := \text{BorderPolynomial}(T_i, \emptyset)$ 
4   solve  $\exists \mathbf{y}(bp_i(\mathbf{u}) \neq 0, T_i(\mathbf{u}, \mathbf{y}) = 0)$ , and let  $Q_i$  be the
   resulting quantifier-free formula
5   if  $Q_i \neq \text{false}$  then output  $[Q_i, T_i, \emptyset]$ 
```

which only involves equations, is obtained by the above algorithm.

Proof of Algorithm 1. The termination of the algorithm is obvious. Let us prove its correctness. Let $R_i = [Q_i, T_i, \emptyset]$, for $i = 1 \dots t$ be the output of Algorithm 1 and let T_j for $j = t + 1 \dots s$ be the regular chains such that $Q_j = \text{false}$. By Lemma 3, each R_i is a regular semi-algebraic system. For $i = 1 \dots s$, define $F_i = \text{sat}(T_i)$. Then we have $V(F) = \cup_{i=1}^s V(F_i)$, where each F_i is equidimensional. For each $i = 1 \dots s$, by Proposition 1, we have

$$V(F_i) \setminus V(bp_i) = V(T_i) \setminus V(bp_i).$$

Moreover, we have

$$V(F_i) = (V(F_i) \setminus V(bp_i)) \cup V(F_i \cup \{bp_i\}).$$

Hence,

$$Z_{\mathbb{R}}(R_i) = Z_{\mathbb{R}}(T_i) \setminus Z_{\mathbb{R}}(bp_i) \subseteq Z_{\mathbb{R}}(F_i) \subseteq Z_{\mathbb{R}}(F)$$

holds. In addition, since bp_i is regular modulo F_i , we have

$$\begin{aligned} Z_{\mathbb{R}}(F) \setminus \cup_{i=1}^t Z_{\mathbb{R}}(R_i) &= \cup_{i=1}^s Z_{\mathbb{R}}(F_i) \setminus \cup_{i=1}^t Z_{\mathbb{R}}(R_i) \\ &\subseteq \cup_{i=1}^s Z_{\mathbb{R}}(F_i) \setminus (Z_{\mathbb{R}}(T_i) \setminus Z_{\mathbb{R}}(bp_i)) \\ &\subseteq \cup_{i=1}^s Z_{\mathbb{R}}(F_i \cup \{bp_i\}), \end{aligned}$$

and $\dim(\cup_{i=1}^s V(F_i \cup \{bp_i\})) < \dim(V(F))$. So the R_i , for $i = 1 \dots t$, form a lazy triangular decomposition of $\mathfrak{S}$. $\square$

In this section, under some genericity assumptions for F , we establish running time estimates for Algorithm 1, see Proposition 3. This is achieved through:

- (1) Proposition 2 giving running time and output size estimates for a Kalkbrener triangular decomposition of an algebraic set, and
- (2) Theorem 2 giving running time and output size estimates for a border polynomial computation.

Our assumptions for these results are the following:

- (H₀) $V(F)$ is equidimensional of dimension d ,
- (H₁) $x_1, \dots, x_d$ are algebraically independent modulo each associated prime ideal of the ideal generated by F in $\mathbb{Q}[\mathbf{x}]$,

- (H₂) F consists of $m := n - d$ polynomials, $f_1, \dots, f_m$.

Hypotheses (H₀) and (H₁) are equivalent to the existence of regular chains $T_1, \dots, T_e$ of $\mathbb{Q}[x_1, \dots, x_n]$ such that $x_1, \dots, x_d$ are free w.r.t. each of $T_1, \dots, T_e$ and such that we have $V(F) = \overline{W(T_1)} \cup \dots \cup \overline{W(T_e)}$.

Denote by δ, h respectively the maximum total degree and height of $f_1, \dots, f_m$. In her PhD Thesis [22], Á. Szántó describes an algorithm which computes a Kalkbrener triangular decomposition, $T_1, \dots, T_e$, of $V(F)$. Under Hypotheses (H₀) to (H₂), this algorithm runs in time $m^{O(1)}(\delta^{O(n^2)})^{d+1}$ counting operations in $\mathbb{Q}$, while the total degrees of the polynomials in the output are bounded by $n\delta^{O(m^2)}$. In addition,

$T_1, \dots, T_e$ are square free, *strongly normalized* [18] and *reduced* [1].

From $T_1, \dots, T_e$, we obtain regular chains $E_1, \dots, E_e$ forming another Kalkbrener triangular decomposition of $V(F)$, as follows. Let $i = 1 \dots e$ and $j = (d+1) \dots n$. Let $t_{i,j}$ be the polynomial of T_i with x_j as main variable. Let $e_{i,j}$ be the primitive part of $t_{i,j}$ regarded as a polynomial in $\mathbb{Q}[x_1, \dots, x_d][x_{d+1}, \dots, x_n]$. Define $E_i = \{e_{i,d+1}, \dots, e_{i,n}\}$. According to the complexity results for polynomial operations stated at the beginning of this section, this transformation can be done within $\delta^{O(m^4)O(n)}$ operations in $\mathbb{Q}$.

Dividing $e_{i,j}$ by its initial we obtain a monic polynomial $d_{i,j}$ of $\mathbb{Q}(x_1, \dots, x_d)[x_{d+1}, \dots, x_n]$. Denote by D_i the regular chain $\{d_{i,d+1}, \dots, d_{i,n}\}$. Observe that D_i is the reduced lexicographic Gröbner basis of the radical ideal it generates in $\mathbb{Q}(x_1, \dots, x_d)[x_{d+1}, \dots, x_n]$. So Theorem 1 in [12] applies to each regular chain D_i . For each polynomial $d_{i,j}$, this theorem provides height and total degree estimates expressed as functions of the *degree* [7] and the *height* [19, 16] of the algebraic set $\overline{W(D_i)}$. Note that the degree and height of $\overline{W(D_i)}$ are upper bounded by those of $V(F)$. Write $d_{i,j} = \sum_{\mu} \frac{\alpha_{\mu}}{\beta_{\mu}} \mu$ where each $\mu \in \mathbb{Q}[x_{d+1}, \dots, x_n]$ is a monomial and $\alpha_{\mu}, \beta_{\mu}$ are in $\mathbb{Q}[x_1, \dots, x_d]$ such that $\gcd(\alpha_{\mu}, \beta_{\mu}) = 1$ holds. Let γ be the lcm of the β_{μ} 's. Then for γ and each α_{μ} :

- the total degree is bounded by $2\delta^{2m}$ and,
- the height by $O(\delta^{2m}(m\bar{h} + dm \log(\delta) + n \log(n)))$.

Multiplying $d_{i,j}$ by γ brings $e_{i,j}$ back. We deduce the height and total degree estimates for each $e_{i,j}$ below.

PROPOSITION 2. *The Kalkbrener triangular decomposition $E_1, \dots, E_e$ of $V(F)$ can be computed in $\delta^{O(m^4)O(n)}$ operations in $\mathbb{Q}$. In addition, every polynomial $e_{i,j}$ has total degree upper bounded by $4\delta^{2m} + \delta^m$, and has height upper bounded by $O(\delta^{2m}(m\bar{h} + dm \log(\delta) + n \log(n)))$.*

Next we estimate the running time and output size for computing the border polynomial of a regular system.

THEOREM 2. *Let $R = [T, P]$ be a squarefree regular system of $\mathbb{Q}[\mathbf{u}, \mathbf{y}]$, with $m = \#T$ and $\ell = \#P$. Let bp be the border polynomial of R . Denote by $\delta_R, \bar{h}_R$ respectively the maximum total degree and height of a polynomial in R . Then the total degree of bp is upper bounded by $(\ell + m)2^{m-1}\delta_R^m$, and bp can be computed within $(n\ell + nm)^{O(n)}(2\delta_R)^{O(n)O(m)}\bar{h}_R^3$ bit operations.*

PROOF. Define $G := P \cup \{\text{der}(t) \mid t \in T\}$. We need to compute the $\ell + m$ iterated resultants $\text{res}(g, T)$, for all $g \in G$. Let $g \in G$. Observe that the total degree and height of g are bounded by δ_R and $\bar{h}_R + \log(\delta_R)$ respectively. Define $r_{m+1} := g, \dots, r_i := \text{res}(t_i, r_{i+1}, y_i), \dots, r_1 := \text{res}(t_1, r_2, y_1)$. Let $i \in \{1, \dots, m\}$. Denote by δ_i and $\bar{h}_i$ the total degree and height of r_i , respectively. Using the complexity estimates stated at the beginning of this section, we have $\delta_i \leq 2^{m-i+1}\delta_R^{m-i+2}$ and $\bar{h}_i \leq 2\delta_{i+1}(\bar{h}_{i+1} + n \log(\delta_{i+1} + 1))$. Therefore, we have $\bar{h}_i \leq (2\delta_R)^{O(m^2)}n^{O(m)}\bar{h}_R$. From these size estimates, one can deduce that each resultant r_i (thus the iterated resultants) can be computed within $(2\delta_R)^{O(mn)+O(m^2)}n^{O(m)}\bar{h}_R^2$ bit operations, by the complexity of computing a determinant stated at the beginning of this section.

Hence, the product of all iterated resultants has total degree and height bounded by $(\ell + m)2^{m-1}\delta_R^m$ and $(\ell + m)(2\delta_R)^{O(m^2)}n^{O(m)}\bar{h}_R$, respectively. Thus, the primitive

and squarefree part of this product can be computed within $(n\ell + nm)^{O(n)}(2\delta_R)^{O(n)O(m)}\bar{h}_R^3$ bit operations, based on the complexity of a polynomial gcd computation stated at the beginning of this section. $\square$

PROPOSITION 3. *From the Kalkbrener triangular decomposition $E_1, \dots, E_e$ of Proposition 2, a lazy triangular decomposition of $f_1 = \dots = f_m = 0$ can be computed in $(\delta^{n^2}n^4)^{O(n^2)}\bar{h}^{O(1)}$ bit operations. Thus, a lazy triangular decomposition of this system is computed from the input polynomials in singly exponential time w.r.t. n , counting operations in $\mathbb{Q}$.*

PROOF. For each $i \in \{1 \dots e\}$, let bp_i be the border polynomial of $[E_i, \emptyset]$ and let $\bar{h}_{R_i}$ (resp. δ_{R_i}) be the height (resp. the total degree) bound of the polynomials in the pre-regular semi-algebraic system $R_i = [\{bp_i\}_{\neq}, E_i, \emptyset]$. According to Algorithm 1, the remaining task is to solve the QE problem $\exists \mathbf{y}(bp_i(\mathbf{u}) \neq 0, E_i(\mathbf{u}, \mathbf{y}) = 0)$ for each $i \in \{1 \dots e\}$, which can be solved within $((m+1)\delta_{R_i})^{O(dm)}\bar{h}_{R_i}^{O(1)}$ bit operations, based on the results of [20]. The conclusion follows from the size estimates in Proposition 2 and Theorem 2. $\square$

5. QUANTIFIER ELIMINATION BY RRC

In the last two sections, we saw that in order to compute a triangular decomposition of a semi-algebraic system, a key step is to solve the following quantifier elimination problem:

$$\exists \mathbf{y}(B(\mathbf{u}) \neq 0, T(\mathbf{u}, \mathbf{y}) = 0, P(\mathbf{u}, \mathbf{y}) > 0), \quad (1)$$

where $[B_{\neq}, T, P_{>}]$ is a pre-regular semi-algebraic system of $\mathbb{Q}[\mathbf{u}, \mathbf{y}]$. This problem is an instance of the so-called *real root classification* (RRC) [27]. In this section, we show how to solve this problem when B is what we call a *fingerprint polynomial set*.

Fingerprint polynomial set. Let $R := [B_{\neq}, T, P_{>}]$ be a pre-regular semi-algebraic system of $\mathbb{Q}[\mathbf{u}, \mathbf{y}]$. Let $D \subset \mathbb{Q}[\mathbf{u}]$. Let dp be the product of all polynomials in D . We call D a *fingerprint polynomial set* (FPS) of R if:

- (i) for all $\alpha \in \mathbb{R}^d$, for all $b \in B$ we have:
 $dp(\alpha) \neq 0 \implies b(\alpha) \neq 0$,
- (ii) for all $\alpha, \beta \in \mathbb{R}^d$ with $\alpha \neq \beta$ and $dp(\alpha) \neq 0, dp(\beta) \neq 0$, if the signs of $p(\alpha)$ and $p(\beta)$ are the same for all $p \in D$, then $R(\alpha)$ has real solutions if and only if $R(\beta)$ does.

Hereafter, we present a method to construct an FPS based on projection operators of CAD.

Open projection operator [21, 4]. Hereafter in this section, let $\mathbf{u} = u_1 < \dots < u_d$ be ordered variables. Let $p \in \mathbb{Q}[\mathbf{u}]$ be non-constant. Denote by $\text{factor}(p)$ the set of the non-constant irreducible factors of p . For $A \subset \mathbb{Q}[\mathbf{u}]$, define $\text{factor}(A) = \cup_{p \in A} \text{factor}(p)$. Let C_d (resp. C_0) be the set of the polynomials in $\text{factor}(p)$ with main variable equal to (resp. less than) u_d . The *open projection operator* (*oproj*) w.r.t. variable u_d maps p to a set of polynomials of $\mathbb{Q}[u_1, \dots, u_{d-1}]$ defined below:

$$\begin{aligned} \text{oproj}(p, u_d) &:= C_0 \cup \bigcup_{f, g \in C_d, f \neq g} \text{factor}(\text{res}(f, g, u_d)) \\ &\quad \cup \bigcup_{f \in C_d} \text{factor}(\text{init}(f, u_d) \cdot \text{discrim}(f, u_d)). \end{aligned}$$

Then, we define $\text{oproj}(A, u_d) := \text{oproj}(\Pi_{p \in A} p, u_d)$.

Augmentation. Let $A \subset \mathbb{Q}[\mathbf{u}]$ and $x \in \{u_1, \dots, u_d\}$. Denote by $\text{der}(A, x)$ the *derivative closure* of A w.r.t. x , that is, $\text{der}(A, x) := \cup_{p \in A} \{\text{der}^{(i)}(p, x) \mid 0 \leq i < \deg(p, x)\}$. The

open augmented projected factors of A is denoted by $\text{oaf}(A)$ and defined as follows. Let k be the smallest positive integer such that $A \subset \mathbb{Q}[u_1, \dots, u_k]$ holds. Denote by C the set $\text{factor}(\text{der}(A, u_k))$; we have

- if $k = 1$, then $\text{oaf}(A) := C$;
- if $k > 1$, then $\text{oaf}(A) := C \cup \text{oaf}(\text{oproj}(C, u_k))$.

THEOREM 3. *Let $A \subset \mathbb{Q}[\mathbf{u}]$ be finite and let σ be a map from $\text{oaf}(A)$ to the set of signs $\{-1, +1\}$. Then the set $S_d := \cap_{p \in \text{oaf}(A)} \{u \in \mathbb{R}^d \mid p(u)\sigma(p) > 0\}$ is either empty or a connected open set in $\mathbb{R}^d$.*

PROOF. By induction on d . When $d = 1$, the conclusion follows from Thom's Lemma [2]. Assume $d > 1$. If d is not the smallest positive integer k such that $A \subset \mathbb{Q}[u_1, \dots, u_k]$ holds, then S_d can be written $S_{d-1} \times \mathbb{R}$ and the conclusion follows by induction. Otherwise, write $\text{oaf}(A)$ as $C \cup E$, where $C = \text{factor}(\text{der}(A, u_d))$ and $E = \text{oaf}(\text{oproj}(C, u_d))$. We have: $E \subset \mathbb{Q}[u_1, \dots, u_{d-1}]$. Denote by M the set $\cap_{p \in E} \{u \in \mathbb{R}^{d-1} \mid p(u)\sigma(p) > 0\}$. If M is empty then so is S_d and the conclusion is clear. From now on assume M not empty. Then, by induction hypothesis, M is a connected open set in $\mathbb{R}^{d-1}$. By the definition of the operator oproj , the product of the polynomials in C is delineable over M w.r.t. u_d . Moreover, C is derivative closed (may be empty) w.r.t. u_d . Therefore $\cap_{p \in \text{oaf}(A)} \{u \in \mathbb{R}^d \mid p(u)\sigma(p) > 0\} \subset M \times \mathbb{R}$ is either empty or a connected open set by Thom's Lemma. $\square$

THEOREM 4. *Let $R := [B_{\neq}, T, P_{>}]$ be a pre-regular semi-algebraic system of $\mathbb{Q}[\mathbf{u}, \mathbf{y}]$. The polynomial set $\text{oaf}(B)$ is a fingerprint polynomial set of R .*

PROOF. Recall that the border polynomial bp of $[T, P]$ divides the product of the polynomials in B . We have $\text{factor}(B) \subseteq \text{oaf}(B)$. So $\text{oaf}(B)$ satisfies (i) in the definition of FPS. Let us prove (ii). Let dp be the product of the polynomials in $\text{oaf}(B)$. Let $\alpha, \beta \in \mathbb{R}^d$ such that both $dp(\alpha) \neq 0$, $dp(\beta) \neq 0$ hold and the signs of $p(\alpha)$ and $p(\beta)$ are equal for all $p \in \text{oaf}(B)$. Then, by Theorem 3, α and β belong to the same connected component of $dp(\mathbf{u}) \neq 0$, and thus to the same connected component of $B(\mathbf{u}) \neq 0$. Therefore the number of real solutions of $R(\alpha)$ and that of $R(\beta)$ are the same by Lemma 2. $\square$

From now on, let us assume that the set B in the pre-regular semi-algebraic system $R = [B_{\neq}, T, P_{>}]$ is an FPS of R . We solve the quantifier elimination problem (1) in three steps: (s_1) compute at least one sample point in each connected component of the semi-algebraic set defined by $B(\mathbf{u}) \neq 0$; (s_2) for each sample point α such that the specialized system $R(\alpha)$ possesses real solutions, compute the sign of $b(\alpha)$ for each $b \in B$; (s_3) generate the corresponding quantifier-free formulas.

In practice, when the set B is not an FPS, one adds some polynomials from $\text{oaf}(B)$, using a heuristic procedure (for instance one by one) until Property (ii) of the definition of an FPS is satisfied. This strategy is implemented in Algorithm 3 of Section 6.

6. IMPLEMENTATION

In this section, we present algorithms for **LazyRealTriangularize** and **RealTriangularize** that we have implemented on top of the **RegularChains** library in MAPLE. We also provide experimental results for test problems which are available at www.orcca.on.ca/~cchen/issac10.txt.

Algorithm 2: GeneratePreRegularSas($\mathfrak{S}$)

Input: a semi-algebraic system $\mathfrak{S} = [F, N_{\geq}, P_{>}, H_{\neq}]$
Output: a set of pre-regular semi-algebraic systems $[B_{i\neq}, T_i, P_{i>}]$, $i = 1 \dots e$, such that
 $Z_{\mathbb{R}}(\mathfrak{S}) = \cup_{i=1}^e Z_{\mathbb{R}}(B_{i\neq}, T_i, P_{i>}) \cup \{\prod_{b \in B_i} b\}, N_{\geq}, P_{>}, H_{\neq}\}.$

```

1  $\mathfrak{T} := \text{Triangularize}(F)$ ;  $\mathfrak{T}' := \emptyset$ 
2 for  $p \in P \cup H$  do
3   for  $T \in \mathfrak{T}$  do
4     for  $C \in \text{Regularize}(p, T)$  do
5       if  $p \notin \text{sat}(C)$  then  $\mathfrak{T}' := \mathfrak{T}' \cup \{C\}$ 
6    $\mathfrak{T} := \mathfrak{T}'; \mathfrak{T}' := \emptyset$ 
7  $\mathfrak{T} := \{[T, \emptyset] \mid T \in \mathfrak{T}\}; \mathfrak{T}' := \emptyset$ 
8 for  $p \in N$  do
9   for  $[T, N'] \in \mathfrak{T}$  do
10    for  $C \in \text{Regularize}(p, T)$  do
11      if  $p \in \text{sat}(C)$  then
12         $\mathfrak{T}' := \mathfrak{T}' \cup \{[C, N']\}$ 
13      else
14         $\mathfrak{T}' := \mathfrak{T}' \cup \{[C, N' \cup \{p\}]\}$ 
15     $\mathfrak{T} := \mathfrak{T}'; \mathfrak{T}' := \emptyset$ 
16  $\mathfrak{T} := \{[T, N', P, H] \mid [T, N'] \in \mathfrak{T}\}$ 
17 for  $[T, N', P, H] \in \mathfrak{T}$  do
18    $B := \text{BorderPolynomialSet}(T, N' \cup P \cup H)$ 
19   output  $[B, T, N' \cup P]$ 
```

Algorithm 3: GenerateRegularSas(B, T, P)

Input: $\mathfrak{S} = [B_{\neq}, T, P_{>}]$, a pre-regular semi-algebraic system of $\mathbb{Q}[\mathbf{u}, \mathbf{y}]$, where $\mathbf{u} = u_1, \dots, u_d$ and $\mathbf{y} = y_1, \dots, y_{n-d}$.
Output: A pair $(D, \mathcal{R})$ satisfying:
(1) $D \subset \mathbb{Q}[\mathbf{u}]$ such that $\text{factor}(B) \subseteq D$;
(2) $\mathcal{R}$ is a finite set of regular semi-algebraic systems, s.t. $\cup_{R \in \mathcal{R}} Z_{\mathbb{R}}(R) = Z_{\mathbb{R}}(D_{\neq}, T, P_{>})$.

```

1  $D := \text{factor}(B \setminus \mathbb{Q})$ 
2 if  $d = 0$  then
3   if  $\text{RealRootCounting}(T, P) = 0$  then
4     return  $(D, \emptyset)$ 
5   else
6     return  $(D, \{\{true, T, P\}\})$ 
7 while true do
8    $S := \text{SamplePoints}(D, d); G_0 := \emptyset; G_1 := \emptyset$ 
9   for  $s \in S$  do
10    if  $\text{RealRootCounting}(T(s), P(s)) = 0$  then
11       $G_0 := G_0 \cup \{\text{GenerateFormula}(D, s)\}$ 
12    else
13       $G_1 := G_1 \cup \{\text{GenerateFormula}(D, s)\}$ 
14  if  $G_0 \cap G_1 = \emptyset$  then
15     $Q := \text{Disjunction}(G_1)$ 
16    if  $Q = false$  then return  $(D, \emptyset)$ 
17    else return  $(D, \{\{Q, T, P\}\})$ 
18  else
19    select a subset  $D' \subseteq \text{oaf}(B) \setminus D$  by some
    heuristic method
20     $D := D \cup D'$ 
```

Basic subroutines. For a zero-dimensional squarefree regular system $[T, P]$, $\text{RealRootCounting}(T, P)$ [23] returns the number of real zeros of $[T, P_>]$. For $A \subset \mathbb{Q}[u_1, \dots, u_d]$ and a point s of $\mathbb{Q}^d$ such that $p(s) \neq 0$ for all $p \in A$, $\text{GenerateFormula}(A, s)$ computes a formula $\bigwedge_{p \in A} (p \sigma_{p,s} > 0)$, where $\sigma_{p,s}$ is defined as +1 if $p(s) > 0$ and -1 otherwise. For a set of formulas G , $\text{Disjunction}(G)$ computes a logic formula Φ equivalent to the disjunction of the formulas in G .

Proof of Algorithm 2. Its termination is obvious. Let us prove its correctness. By the specification of Triangularize and Regularize , at line 16, we have

$$Z(F, P_{\neq} \cup H_{\neq}) = \cup_{[T, N', P, H] \in \mathfrak{T}} Z(\text{sat}(T), P_{\neq} \cup H_{\neq}).$$

Write $\cup_{[T, N', P, H] \in \mathfrak{T}}$ as $\cup_T$. Then we deduce that

$$Z_{\mathbb{R}}(F, N_{\geq}, P_{>}, H_{\neq}) = \cup_T Z_{\mathbb{R}}(\text{sat}(T), N_{\geq}, P_{>}, H_{\neq}).$$

For each $[T, N', P, H]$, at line 19, we generate a pre-regular semi-algebraic system $[B_{\neq}, T, N'_{>} \cup P_{>}]$. By Proposition 1, we have

$$\begin{aligned} Z_{\mathbb{R}}(\text{sat}(T), N_{\geq}, P_{>}, H_{\neq}) = \\ Z_{\mathbb{R}}(B_{\neq}, T, N'_{>} \cup P_{>}) \cup Z_{\mathbb{R}}(\text{sat}(T) \cup \{\Pi_{b \in B} b\}, N_{\geq}, P_{>}, H_{\neq}), \end{aligned}$$

which implies that

$$\begin{aligned} Z_{\mathbb{R}}(\mathfrak{S}) = & \cup_T Z_{\mathbb{R}}(B_{\neq}, T, N'_{>} \cup P_{>}) \\ & \cup_T Z_{\mathbb{R}}(\text{sat}(T) \cup \{\Pi_{b \in B} b\}, N_{\geq}, P_{>}, H_{\neq}). \end{aligned}$$

So Algorithm 2 satisfies its specification.

Algorithm 4: SamplePoints(A, k)

Input: $A \subset \mathbb{Q}[x_1, \dots, x_k]$ is a finite set of non-zero polynomials
Output: A finite subset of $\mathbb{Q}^k$ contained in $(\Pi_{p \in A} p) \neq 0$ and having a non-empty intersection with each connected component of $(\Pi_{p \in A} p) \neq 0$.
1 **if** $k = 1$ **then**
2 return one rational point from each connected component of $\Pi_{p \in A} p \neq 0$
3 **else**
4 $A_k := \{p \in A \mid \text{mvar}(p) = x_k\}$; $A' := \text{oproj}(A, x_k)$
5 **for** $s \in \text{SamplePoints}(A', k-1)$ **do**
6 Collect in a set S one rational point from each connected component of $\Pi_{p \in A_k} p(s, x_k) \neq 0$;
7 **for** $\alpha \in S$ **do** output (s, α)

Algorithm 5: LazyRealTriangularize($\mathfrak{S}$)

Input: a semi-algebraic system $\mathfrak{S} = [F, N_{\geq}, P_{>}, H_{\neq}]$
Output: a lazy triangular decomposition of $\mathfrak{S}$
1 $\mathfrak{T} := \text{GeneratePreRegularSas}(F, N, P, H)$
2 **for** $[B, T, P'] \in \mathfrak{T}$ **do**
3 $(D, \mathcal{R}) = \text{GenerateRegularSas}(B, T, P')$
4 **if** $\mathcal{R} \neq \emptyset$ **then** output $\mathcal{R}$

Proof of Algorithms 3 and 4. By the definition of oproj , Algorithm 4 terminates and satisfies its specification. By Theorem 4, $\text{oaf}(B)$ is an FPS. Thus, by the definition of an FPS, Algorithm 3 terminates and satisfies its specification.

Proof of Algorithm 5. Its termination is obvious. Let us prove the algorithm is correct. Let R_i , $i = 1 \dots t$ be the

Algorithm 6: RealTriangularize($\mathfrak{S}$)

Input: a semi-algebraic system $\mathfrak{S} = [F, N_{\geq}, P_{>}, H_{\neq}]$
Output: a triangular decomposition of $\mathfrak{S}$
1 $\mathfrak{T} := \text{GeneratePreRegularSas}(F, N, P, H)$
2 **for** $[B, T, P'] \in \mathfrak{T}$ **do**
3 $(D, \mathcal{R}) = \text{GenerateRegularSas}(B, T, P')$
4 **if** $\mathcal{R} \neq \emptyset$ **then** output $\mathcal{R}$
5 **for** $p \in D$ **do**
6 output $\text{RealTriangularize}(F \cup \{p\}, N, P, H)$

output. By the specification of each sub-algorithm, each R_i is a regular semi-algebraic system and we have:

$$\cup_{i=1}^t Z_{\mathbb{R}}(R_i) \subseteq Z_{\mathbb{R}}(\mathfrak{S}).$$

Next we show that there exists an ideal $\mathcal{I} \subseteq \mathbb{Q}[\mathbf{x}]$, whose dimension is less than $\dim(Z(F, P_{\neq} \cup H_{\neq}))$ and such that $Z_{\mathbb{R}}(\mathfrak{S}) \setminus \cup_{i=1}^t Z_{\mathbb{R}}(R_i) \subseteq Z_{\mathbb{R}}(\mathcal{I})$ holds.

At line 1, by the specification of Algorithm 2, we have

$$\begin{aligned} Z_{\mathbb{R}}(\mathfrak{S}) = & \cup_T Z_{\mathbb{R}}(B_{\neq}, T, P'_{>}) \\ & \cup_T Z_{\mathbb{R}}(\text{sat}(T) \cup \{\Pi_{b \in B} b\}, N_{\geq}, P_{>}, H_{\neq}). \end{aligned}$$

At line 3, by the specification of Algorithm 3, for each B , we compute a set D such that $\text{factor}(B) \subseteq D$ and

$$\cup_T Z_{\mathbb{R}}(D_{\neq}, T, P'_{>}) = \cup_{i=1}^t Z_{\mathbb{R}}(R_i)$$

both hold. Combining the two relations together, we have

$$\begin{aligned} Z_{\mathbb{R}}(\mathfrak{S}) = & \cup_T Z_{\mathbb{R}}(R_i) \\ & \cup_T Z_{\mathbb{R}}(\text{sat}(T) \cup \{\Pi_{p \in D} p\}, N_{\geq}, P_{>}, H_{\neq}). \end{aligned}$$

Therefore, the following relations hold

$$\begin{aligned} Z_{\mathbb{R}}(\mathfrak{S}) \setminus \cup_{i=1}^t Z_{\mathbb{R}}(R_i) \\ \subseteq \cup_T Z_{\mathbb{R}}(\text{sat}(T) \cup \{\Pi_{p \in D} p\}, N_{\geq}, P_{>}, H_{\neq}) \\ \subseteq Z_{\mathbb{R}}(\cap_T (\text{sat}(T) \cup \{\Pi_{p \in D} p\})). \end{aligned}$$

Define

$$\mathcal{I} = \cap_T (\text{sat}(T) \cup \{\Pi_{p \in D} p\}).$$

Since each $p \in D$ is regular modulo $\text{sat}(T)$, we have

$$\dim(\mathcal{I}) < \dim(\cap_T \text{sat}(T)) \leq \dim(Z(F, P_{\neq} \cup H_{\neq})).$$

So all R_i form a lazy triangular decomposition of $\mathfrak{S}$. $\square$

Proof of Algorithm 6. For its termination, it is sufficient to prove that there are only finitely many recursive calls to RealTriangularize . Indeed, if $[F, N, P, H]$ is the input of a call to RealTriangularize then each of the immediate recursive calls takes $[F \cup \{p\}, N, P, H]$ as input, where p belongs to the set D of some pre-regular semi-algebraic system $[D_{\neq}, T, P_{>}]$. Since p is regular (and non-zero) modulo $\text{sat}(T)$ we have:

$$\langle F \rangle \subsetneq \langle F \cup \{p\} \rangle.$$

Therefore, the algorithm terminates by the ascending chain condition on ideals of $\mathbb{Q}[\mathbf{x}]$. The correctness of Algorithm 6 follows from the specifications of the sub-algorithms. $\square$

Table 1. Table 1 summarizes the notations used in Tables 2 and 3. Tables 2 and 3 demonstrate benchmarks running in MAPLE 14 β 1, using an Intel Core 2 Quad CPU (2.40GHz) with 3.0GB memory. The timings are in seconds and the time-out is 1 hour.

Table 2. The systems in this group involve equations only. We report the running times for a triangular decomposition

Table 1 Notations for Tables 2 and 3

symbol	meaning
#e	number of equations in the input system
#v	number of variables in the input equations
d	maximum total degree of an input equation
G	Groebner-Basis (plex order) in MAPLE
T	Triangularize in RegularChains library of MAPLE
LR	LazyRealTriangularize implemented in MAPLE
R	RealTriangularize implemented in MAPLE
Q	QEPCAD B
> 1h	computation does not complete within 1 hour
FAIL	QEPCAD B failed due to prime list exhausted

Table 2 Timings for varieties

system	#v/#e/d	G	T	LR
Hairer-2-BGK	13/ 11/ 4	25	1.924	2.396
Collins-jsc02	5/ 4/ 3	876	0.296	0.820
Leykin-1	8/ 6/ 4	103	3.684	3.924
8-3-config-Li	12/ 7/ 2	109	5.440	6.360
Lichtblau	3/ 2/ 11	126	1.548	11
Cinquin-3-3	4/ 3/ 4	64	0.744	2.016
Cinquin-3-4	4/ 3/ 5	> 1h	10	22
DonatiTraverso-rev	4/ 3/ 8	154	7.100	7.548
Cheaters-homotopy-1	7/ 3/ 7	3527	174	> 1h
hereman-8.8	8/ 6/ 6	> 1h	33	62
L	12/ 4/ 3	> 1h	0.468	0.676
dgp6	17/19/ 2	27	60	63
dgp29	5/ 4/ 15	84	0.008	0.016

of the input algebraic variety and a lazy triangular decomposition of the corresponding real variety. These illustrate the good performance of our tool.

Table 3. The examples in this table are quantifier elimination problems and most of them involve both equations and inequalities. We provide the timings for computing a lazy and a full triangular decomposition of the corresponding semi-algebraic system and the timings for solving the quantifier elimination problem via QEPCAD B [5] (in non-interactive mode). Computations complete with our tool on more examples than with QEPCAD B.

Remark. The output of our tools is a set of regular semi-algebraic systems, which is different than that of QEPCAD B. We note also that our tool is more effective for systems with more equations than inequalities.

Acknowledgments. The authors would like to thank the referees for their valuable remarks that helped to improve the presentation of the work.

Table 3 Timings for semi-algebraic systems

system	#v/#e/d	T	LR	R	Q
BM05-1	4/ 2/ 3	0.008	0.208	0.568	86
BM05-2	4/ 2/ 4	0.040	2.284	> 1h	FAIL
Solotareff-4b	5/ 4/ 3	0.640	2.248	924	> 1h
Solotareff-4a	5/ 4/ 3	0.424	1.228	8.216	FAIL
putnam	6/ 4/ 2	0.044	0.108	0.948	> 1h
MPV89	6/ 3/ 4	0.016	0.496	2.544	> 1h
IBVP	8/ 5/ 2	0.272	0.560	12	> 1h
Lafferriere37	3/ 3/ 4	0.056	0.184	0.180	10
Xia	6/ 3/ 4	0.164	191	739	> 1h
SEIT	11/ 4/ 3	0.400	> 1h	> 1h	> 1h
p3p-isosceles	7/ 3/ 3	1.348	> 1h	> 1h	> 1h
p3p	8/ 3/ 3	210	> 1h	> 1h	FAIL
Ellipse	6/ 1/ 3	0.012	> 1h	> 1h	> 1h

7. REFERENCES

- [1] P. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. *J. Symb. Comput.*, 28(1-2):105–124, 1999.
- [2] S. Basu, R. Pollack, and M-F. Roy. *Algorithms in real algebraic geometry*. Springer-Verlag, 2006.
- [3] F. Boulrier, C. Chen, F. Lemaire, and M. Moreno Maza. Real root isolation of regular chains. In *Proc. ASCM'09*.
- [4] C. W. Brown. Improved projection for cylindrical algebraic decomposition. *J. Symb. Comput.*, 32(5):447–465, 2001.
- [5] C. W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using cads. *SIGSAM Bull.*, 37(4):97–108, 2003.
- [6] C. W. Brown and S. McCallum. On using bi-equational constraints in cad construction. In *ISSAC'05*, pages 76–83, 2005.
- [7] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic Complexity Theory*. Springer, 1997.
- [8] C. Chen, O. Golubitsky, F. Lemaire, M. Moreno Maza, and W. Pan. Comprehensive triangular decomposition. In *CASC'07*, pages 73–101, 2007.
- [9] C. Chen, M. Moreno Maza, B. Xia, and L. Yang. Computing cylindrical algebraic decomposition via triangular decomposition. In *ISSAC'09*, pages 95–102.
- [10] J.S. Cheng, X.S. Gao, and C.K. Yap. Complete numerical isolation of real zeros in zero-dimensional triangular systems. In *ISSAC '07*, pages 92–99, 2007.
- [11] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. *Springer Lecture Notes in Computer Science*, 33:515–532, 1975.
- [12] X. Dahan, A. Kadri, and É. Schost. Bit-size estimates for triangular sets in positive dimension. Technical report, University of Western Ontario, 2009.
- [13] X. Dahan, M. Moreno Maza, É. Schost, W. Wu, and Y. Xie. Lifting techniques for triangular decompositions. In *ISSAC'05*, pages 108–115, 2005.
- [14] J.H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra*. Academic Press, 1988.
- [15] H. Hong and J. R. Sendra. Computation of variant results, B. Caviness and J. Johnson, eds, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, 1998.
- [16] T. Krick, L. M. Pardo, and M. Sombra. Sharp estimates for the arithmetic Nullstellensatz. *Duke Math. J.*, 109(3):521–598, 2001.
- [17] X. Li, M. Moreno Maza, and W. Pan. Computations modulo regular chains. In *ISSAC'09*, pages 239–246, 2009.
- [18] M. Moreno Maza. On triangular decompositions of algebraic varieties. MEGA-2000, Bath, UK. <http://www.csd.uwo.ca/~moreno/books-papers.html>
- [19] P. Philippon. Sur des hauteurs alternatives III. *J. Math. Pures Appl.*, 74(4):345–365, 1995.
- [20] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. parts I–III. *J. Symb. Comput.*, 13(3):255–352, 1992.
- [21] A. Strzeboński. Solving systems of strict polynomial inequalities. *J. Symb. Comput.*, 29(3):471–480, 2000.
- [22] Á. Szántó. *Computation with polynomial systems*. PhD thesis, Cornell University, 1999.
- [23] B. Xia and T. Zhang. Real solution isolation using interval arithmetic. *Comput. Math. Appl.*, 52(6-7):853–860, 2006.
- [24] R. Xiao. *Parametric Polynomial System Solving*. PhD thesis, Peking University, Beijing, 2009.
- [25] L. Yang, X. Hou, and B. Xia. A complete algorithm for automated discovering of a class of inequality-type theorems. *Science in China, Series F*, 44(6):33–49, 2001.
- [26] L. Yang and B. Xia. *Automated proving and discovering inequalities*. Science Press, Beijing, 2008.
- [27] L. Yang and B. Xia. Real solution classifications of a class of parametric semi-algebraic systems. In *A3L'05*, pages 281–289, 2005.

When Can We Detect that a P-Finite Sequence is Positive?

Manuel Kauers^{*}

RISC
Johannes Kepler University
4040 Linz (Austria)
mkauers@risc.jku.at

Veronika Pillwein[†]

RISC
Johannes Kepler University
4040 Linz (Austria)
vpillwei@risc.jku.at

ABSTRACT

We consider two algorithms which can be used for proving positivity of sequences that are defined by a linear recurrence equation with polynomial coefficients (P-finite sequences). Both algorithms have in common that while they do succeed on a great many examples, there is no guarantee for them to terminate, and they do in fact not terminate for every input. For some restricted classes of P-finite recurrence equations of order up to three we provide a priori criteria that assert the termination of the algorithms.

Categories and Subject Descriptors

I.1.2 [Computing Methodologies]: Symbolic and Algebraic Manipulation—*Algorithms*; G.2.1 [Discrete Mathematics]: Combinatorics—*Recurrences and difference equations*

General Terms

Algorithms

Keywords

P-finite Sequences, Positivity, Cylindrical decomposition

1. INTRODUCTION

Inequalities for special functions are a serious challenge, both from the traditional paper-and-pencil point of view, but also (and in particular) for computer algebra. In contrast to the vast number of algorithms for dealing with identities, almost no algorithms are available for inequalities. Already for the very restricted class of sequences satisfying linear

recurrence equations with constant coefficients (C-finite sequences), the positivity problem leads to hard number theoretic questions to which no solutions are known today, see [8, 10] and the references given there for the current state of the struggle.

Still, inequalities are not entirely hopeless. For example, Mezzarobba and Salvy have recently given an algorithm for effectively computing tight upper bounds for sequences defined by linear recurrence equations with polynomial coefficients (P-finite sequences) [16]. Five years ago, Gerhold and Kauers [11] proposed a method applicable to inequalities concerning quantities that satisfy recurrence equations of a very general type. Their method consists of constructing a sequence of polynomial sufficient conditions that would imply the non-polynomial inequality under consideration. If one of the conditions in the sequence happens to be true (which can be detected, e.g., with Cylindrical Algebraic Decomposition [5, 6, 4, 2]), the method succeeds, otherwise it keeps on running forever. Simultaneously, the method searches for counterexamples and it will find one and terminate for every false inequality.

Despite its simplicity, the method has proven quite successful in applications. Not only did it provide the first computer proofs of some special function inequalities from the literature [11, 12, 13, 14], but it even helped to resolve some open conjectures [1, 15, 14, 17]. At the same time, the method remains somewhat unsatisfactory from a computational point of view, as it is not clear on which inequalities it succeeds and on which it doesn't. It would be interesting to have, at least for some restricted classes, some a priori criteria telling us whether the method (or some variation of it) will succeed or not.

Our goal in this paper is to provide such criteria for two particular proving procedures (Algorithms 1 and 2 described below). We are far from being able to give a full answer to the question posed in the title, but we can identify some nontrivial portions of P-finite recurrence equations of fixed order on which termination of Algorithms 1 or 2 is guaranteed. For first order equations, deciding positivity is trivial. For second order equations, we provide a result (Theorems 2 and 3) that answers the question under a genericity assumption. For third order equations, we are able to identify the terminating cases of Algorithm 2 but only have partial results for Algorithm 1 supplemented by empirical evidence supporting a conjecture concerning its terminating cases. An interesting aspect of our analysis is that algorithms for real quantifier elimination are not only used as a subroutine of Algorithms 1 and 2, but they are also contributing in an essential way to the proofs of our termination theorems. It

^{*}Supported by the Austrian Science Fund (FWF) grants P20347-N18 and Y464-N18.

[†]Supported by the Austrian Science Fund (FWF) grant W1214/DK6.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007...\$10.00.

is therefore possible—in principle—to extend our results to equations of order greater than three. Only the increasing time and memory requirements of the computations have prevented us from doing so.

2. PRELIMINARIES

A sequence $f: \mathbb{N} \rightarrow K := \mathbb{R} \cap \bar{\mathbb{Q}}$ is called *P-finite* (or *holonomic*) if there exist polynomials $p_0, \dots, p_r \in K[x]$, not all zero, such that

$$p_0(n)f(n) + p_1(n)f(n+1) + \dots + p_r(n)f(n+r) = 0$$

for all $n \in \mathbb{N}$. Such an equation is called a (P-finite) *recurrence*, and r is called its order. If $p_r(n) \neq 0$ for all $n \in \mathbb{N}$, then the infinite sequence f is uniquely determined by the recurrence and r *initial values* $f(0), f(1), \dots, f(r-1)$. The assumption $p_r(n) \neq 0$ for all $n \in \mathbb{N}$ can be adopted without loss of generality, because we can substitute $g(n) = f(n+u)$ for some u larger than the greatest integer root of p_r and then consider g instead of f and check nonnegativity of the finitely many terms $f(0), f(1), \dots, f(u-1)$ by inspection. We will do so.

From now on, all recurrences are assumed to have a leading coefficient p_r with no positive integer roots.

A P-finite recurrence is called *balanced* if $\deg p_0 = \deg p_r$ and $\deg p_i \leq \deg p_0$ ($i = 1, \dots, r$). The *characteristic polynomial* of a balanced recurrence is defined as

$$\text{lc}_y(p_0(y) + p_1(y)x + p_2(y)x^2 + \dots + p_r(y)x^r) \in K[x].$$

Its roots $\lambda_1, \dots, \lambda_r \in \mathbb{C}$ are called the *eigenvalues* of the recurrence. (The λ_i are not necessarily distinct.) Note that for a balanced recurrence, the characteristic polynomial has always degree r and it has never 0 as a root.

An eigenvalue λ_i is called *dominant* if $|\lambda_j| \leq |\lambda_i|$ for all $j = 1, \dots, r$. Dominant eigenvalues govern the asymptotics of the sequences defined by the recurrence [20, 9]. If there is a unique dominant eigenvalue λ_i , then for we will usually have

$$f(n) \sim c(n)\lambda_i^n \quad (n \rightarrow \infty)$$

where c is of subexponential growth in the sense that

$$\frac{c(n+1)}{c(n)} \xrightarrow{n \rightarrow \infty} 1.$$

There may be choices of initial values for which $c(n) = 0$ for all n so that the asymptotics of f is not affected by λ_i but by the next smaller eigenvalue(s). Whether this is the case or not can be hard to verify formally, but is usually easy to verify empirically. Some of our termination results apply only to this *generic* situation where initial values are chosen such as to actually exhibit the asymptotic behavior predicted by the dominant eigenvalue.

Finally, if the dominant eigenvalue λ_i is not real and positive, then it is clear that f will be ultimately oscillating, and so $f(n) \geq 0$ cannot possibly be true for all n . This case can be sorted out trivially beforehand, and we may therefore assume that the unique dominant eigenvalue is real and positive. In this case, the substitution $g(n) = f(n)/\lambda_i^n$ turns the recurrence

$$p_0(n)g(n) + p_1(n)g(n+1) + \dots + p_r(n)g(n+r) = 0$$

into

$$p_0(n)g(n) + p_1(n)\lambda_i g(n+1) + \dots + p_r(n)\lambda_i^r g(n+r) = 0$$

whose dominant eigenvalue is 1. As $g(n) \geq 0 \iff f(n) \geq 0$, it suffices to consider this case.

3. INDUCTION BASED PROVING PROCEDURES

3.1 The Original Version

The approach of [11] is as follows. Suppose that $f: \mathbb{N} \rightarrow K$ is defined by a recurrence

$$p_0(n)f(n) + p_1(n)f(n+1) + \dots + p_r(n)f(n+r) = 0$$

and initial values $f(0) = f_0, f(1) = f_1, \dots, f(r-1) = f_{r-1}$. We seek to prove $f(n) \geq 0$ for all $n \in \mathbb{N}$ by induction:

$$f(n) \geq 0 \wedge \dots \wedge f(n+r-1) \geq 0 \implies f(n+r) \geq 0.$$

Because of the recurrence, this is equivalent to

$$\begin{aligned} f(n) \geq 0 \wedge \dots \wedge f(n+r-1) \geq 0 \\ \implies -\frac{p_0(n)}{p_r(n)}f(n) - \dots - \frac{p_{r-1}(n)}{p_r(n)}f(n+r-1) \geq 0 \end{aligned}$$

For this to be true for all $n \in \mathbb{N}$, it is sufficient that the *induction step formula*

$$\begin{aligned} \forall y_0, y_1, \dots, y_{r-1} \in \mathbb{R} \forall x \in \mathbb{R} : \\ (x \geq 0 \wedge y_0 \geq 0 \wedge \dots \wedge y_{r-1} \geq 0) \\ \implies -\frac{p_0(x)}{p_r(x)}y_0 - \dots - \frac{p_{r-1}(x)}{p_r(x)}y_{r-1} \geq 0 \end{aligned}$$

is true, and this can be decided by a quantifier elimination algorithm. If it is true, the induction step is established and f is nonnegative everywhere if and only if it is nonnegative for $n = 0, \dots, r-1$, which can be checked.

In the unlucky case when the induction step formula is false, there is no immediate conclusion about f that could be drawn. In this case, *refined induction step formulas*

$$f(n) \geq 0 \wedge \dots \wedge f(n+q-1) \geq 0 \implies f(n+q) \geq 0$$

for $q > r$ are constructed. Using the recurrence, each term $f(n+i)$ can be rewritten as a linear combination of $f(n), \dots, f(n+r-1)$ with rational function coefficients, and using this rewriting, the refined induction step formula takes the form

$$\begin{aligned} \Phi(q) := \forall y_0, y_1, \dots, y_{r-1} \in \mathbb{R} \forall x \in \mathbb{R} : \\ (x \geq 0 \wedge y_0 \geq 0 \wedge \dots \wedge y_{r-1} \geq 0 \\ \wedge q_{r,0}(x)y_0 + \dots + q_{r,r-1}(x)y_{r-1} \geq 0 \\ \wedge q_{r+1,0}(x)y_0 + \dots + q_{r+1,r-1}(x)y_{r-1} \geq 0 \\ \vdots \\ \wedge q_{q-1,0}(x)y_0 + \dots + q_{q-1,r-1}(x)y_{r-1} \geq 0) \\ \implies q_{q,0}(x)y_0 + \dots + q_{q,r-1}(x)y_{r-1} \geq 0, \end{aligned}$$

where the $q_{i,j}$ are some rational functions.

The full method then reads as follows.

ALGORITHM 1.

Input: A P-finite recurrence of order r and a vector of initial

values defining a sequence $f: \mathbb{N} \rightarrow K$.

Output: True if $f(n) \geq 0$ for all $n \in \mathbb{N}$, False if $f(n) < 0$ for some $n \in \mathbb{N}$, possibly no output at all.

```

1 for  $n = 0$  to  $r - 1$  do
2   if  $f(n) < 0$  then return False
3 for  $n = r, r + 1, r + 2, r + 3, \dots$  do
4   if  $\Phi(n)$  then return True
5   if  $f(n) < 0$  then return False

```

EXAMPLE 1. Let $f: \mathbb{N} \rightarrow K$ be defined by

$$\begin{aligned}
&(2n + 13)f(n + 3) - (5n + 22)f(n + 2) \\
&\quad + (3n + 20)f(n + 1) - (2n + 7)f(n) = 0, \\
&f(0) = f(1) = f(2) = 1.
\end{aligned}$$

We use Algorithm 1 to show that $f(n) \geq 0$ for all $n \in \mathbb{N}$.

Since $f(0), f(1), f(2) \geq 0$, we enter the loop in line 3. For $n = 3$, we have

$$\begin{aligned}
\Phi(n) &= \forall y_0, y_1, y_2 \forall x \in \mathbb{R} : \\
&(x \geq 0 \wedge y_0 \geq 0 \wedge y_1 \geq 0 \wedge y_2 \geq 0) \\
&\implies \frac{2x+7}{2x+13}y_0 - \frac{3x+20}{2x+13}y_1 + \frac{5x+22}{2x+13}y_2 \geq 0.
\end{aligned}$$

This is false, but since $f(3) = 9/13 > 0$ (checked in line 5), we continue.

The formula $\Phi(4)$ is too lengthy to be reproduced here explicitly, and it is also false. Yet $f(4) = 61/195 \geq 0$, so we proceed to consider the even lengthier formula $\Phi(5)$, which turns out to be true. At this point the algorithm terminates with output True.

3.2 A Variation

In cases where Algorithm 1 does not terminate, it is sometimes possible to prove inductively the stronger statement that $f(n)$ is increasing, viz. that $f(n + 1) \geq f(n)$ for all $n \geq 0$. While this is obviously a sufficient condition for $f(n) \geq 0$ for all n , there are of course sequences f which are non-negative but not increasing. For such cases, a good strategy is to prove that $\mu^{-n}f(n)$ is increasing, for some suitably chosen constant $\mu > 0$. The choice of μ is critical in two respects: it must be small enough to assure that $\mu^{-n}f(n)$ actually is increasing, and it must be big enough to allow for an inductive proof. The following algorithm proves positivity of a P-finite sequence f by searching for a μ that meets both criteria.

ALGORITHM 2.

Input: A P-finite recurrence of order r and a vector of initial values defining a sequence $f: \mathbb{N} \rightarrow K$.

Output: True if $f(n) \geq 0$ for all $n \in \mathbb{N}$, False if $f(n) < 0$ for some $n \in \mathbb{N}$, possibly no output at all.

```

1 Determine a quantifier free formula  $\Phi(\xi, \mu)$  equivalent
  to

```

$$\begin{aligned}
&\forall y_0, \dots, y_{r-1} \forall x \geq \xi : \\
&\left(y_0 \geq 0 \wedge y_1 \geq \mu y_0 \wedge \dots \wedge y_{r-1} \geq \mu y_{r-2} \right) \\
&\implies -\frac{p_0(x)}{p_r(x)}y_0 - \dots - \frac{p_{r-1}(x)}{p_r(x)}y_{r-1} \geq \mu y_{r-1}
\end{aligned}$$

```

2 for  $n = 0, 1, 2, 3, \dots$  do
3   if  $f(n) < 0$  then
4     return False

```

Copyright 2010 ACM 978-1-4503-0150-3/10/0007...\$10.00.

```

5   else if  $\exists \mu \geq 0 : \Phi(n, \mu) \wedge f(n + 1) \geq \mu f(n)$ 
6      $\wedge \dots \wedge f(n + r - 1) \geq \mu f(n + r - 2)$  then
       return True

```

THEOREM 1. Algorithm 2 is correct.

PROOF. Correctness is obvious whenever the algorithm returns False, because this happens only when an explicit point n with $f(n) < 0$ has been found. Suppose now that the algorithm returns True at the n th iteration of the for loop. Then $f(k) \geq 0$ for $k = 0, \dots, n$, otherwise the algorithm would have terminated in an earlier iteration with output False. The condition in line 5 inductively implies

$$\exists \mu \geq 0 \forall k \geq n : f(k + 1) \geq \mu f(k).$$

Since $\mu \geq 0$ and $f(n) \geq 0$, this inductively implies $f(k) \geq 0$ also for all $k > n$. $\square$

EXAMPLE 2. Let $f: \mathbb{N} \rightarrow K$ be defined by

$$\begin{aligned}
&(n + 3)f(n + 3) - (5n + 13)f(n + 2) \\
&\quad + (5n + 12)f(n + 1) - (n + 2)f(n) = 0, \\
&f(0) = 1, f(1) = 1/4, f(2) = 1/10.
\end{aligned}$$

Algorithm 1 does not seem to terminate for this sequence. But Algorithm 2 succeeds.

Step 1 produces the quantifier free formula

$$\xi \geq 0 \wedge \frac{5 - \sqrt{5}}{2} \leq \mu \leq \frac{\sqrt{5\xi^2 + 22\xi + 25} + 5\xi + 13}{2(\xi + 3)},$$

which we denote $\Phi(\xi, \mu)$. In the iteration of the for loop, we get:

For $n = 0$, since $f(0) = 1 \geq 0$, we check whether

$$0 \geq 0 \wedge \frac{5 - \sqrt{5}}{2} \leq \mu \leq 3 \wedge \frac{1}{4} \geq \mu \wedge \frac{1}{10} \geq \frac{\mu}{4}$$

is satisfiable. As it is not, we proceed.

Also $n = 1$ and $n = 2$, we have $f(n) \geq 0$ but there is no $\mu \geq 0$ with

$$\Phi(n, \mu) \wedge f(n + 1) \geq \mu f(n) \wedge f(n + 2) \geq \mu f(n + 1).$$

Then for $n = 3$, since $f(3) \geq 0$, we check whether

$$3 \geq 0 \wedge \frac{5 - \sqrt{5}}{2} \leq \mu \leq \frac{28 + 2\sqrt{34}}{12} \wedge \frac{17}{80} \geq \frac{\mu}{10} \wedge \frac{247}{400} \geq \frac{17\mu}{80}$$

is satisfiable. As it is satisfiable (e.g., by $\mu = 2$) the algorithm terminates with output True.

The two strategies employed in Algorithms 1 and 2 in the case where a direct proof of the induction step formula fails (prolonging the induction hypothesis in Algorithm 1 versus multiplying with a positivity preserving exponential in Algorithm 2) are independent of each other. It is possible to merge both strategies into a single strategy that simultaneously prolongs the induction hypothesis and inserts a positivity preserving exponential. An algorithm based on such a combined strategy is easily written down, but turns out to be computationally quite expensive on examples. It would be interesting to carry out the termination analysis given below for the combined algorithm, but the quantifier elimination problems arising in this analysis seem currently too hard to be carried out for the combined algorithm.

4. TERMINATING CASES

Both algorithms given in the previous section may fail to terminate. Our goal now is to identify classes of P-finite recurrence equations for which termination can be guaranteed a priori.

4.1 Order One

This case is rather simple and included here merely for the sake of completeness. If $f: \mathbb{N} \rightarrow K$ satisfies

$$p_0(n)f(n) + p_1(n)f(n+1) = 0,$$

then $f(n) \geq 0$ for all $n \in \mathbb{N}$ if and only if $f(0) \geq 0$ and $-p_0(n)/p_1(n) \geq 0$ for all $n \in \mathbb{N}$. Since sign changes of $-p_0(n)/p_1(n)$ can occur only at the real roots of p_0 or p_1 , the only thing we need to do is to find an upper bound $n_0 \in \mathbb{R}$ for the real roots (this can be done), and then check whether $-p_0(n)/p_1(n) \geq 0$ for $n = 0, 1, 2, \dots, n_0 + 1$.

EXAMPLE 3. Consider $f: \mathbb{N} \rightarrow K$ defined via

$$(3n - 16)f(n) - (3n - 17)f(n+1) = 0, \quad f(0) = 1.$$

The roots of p_0, p_1 are $16/3$ and $17/3$, respectively, and they are both less than $n_0 = 6$, for instance. Therefore, since $f(n) \geq 0$ for $n = 0, \dots, 6$, we can conclude that $f(n) \geq 0$ for all $n \in \mathbb{N}$.

4.2 Order Two

We now turn to sequences $f: \mathbb{N} \rightarrow K$ which are defined by a balanced P-finite recurrence of second order,

$$p_2(n)f(n+2) - p_1(n)f(n+1) - p_0(n)f(n) = 0.$$

We assume (without loss of generality) that 1 is a dominant eigenvalue of this recurrence and let $u \in K$ with $|u| < 1$ be such that

$$(x-1)(x-u) = x^2 - (u+1)x - (-u)$$

is the characteristic polynomial of the recurrence. The question is whether Algorithm 1 and Algorithm 2 succeed in proving that $f(n) \geq 0$ for all n . (If this is actually the case; if it is not, then both algorithms will obviously succeed in finding a counterexample.) We will show that termination of Algorithm 1 depends on the sign of u whereas Algorithm 2 (generically) terminates for all u .

THEOREM 2. If $u \in (-1, 0)$, then Algorithm 1 terminates.

PROOF. Rewrite the recurrence in the form

$$f(n+2) = \frac{p_1(n)}{p_2(n)}f(n+1) + \frac{p_0(n)}{p_2(n)}f(n).$$

Since the characteristic polynomial is

$$(x-1)(x-u) = x^2 - (u+1)x - (-u),$$

we have

$$\frac{p_1(n)}{p_2(n)} \xrightarrow{n \rightarrow \infty} u+1 > 0 \quad \text{and} \quad \frac{p_0(n)}{p_2(n)} \xrightarrow{n \rightarrow \infty} -u > 0.$$

Therefore,

$$\exists n_0 \in \mathbb{N} \forall n \geq n_0 : \frac{p_1(n)}{p_2(n)} > 0 \wedge \frac{p_0(n)}{p_2(n)} > 0,$$

where we may safely regard n as ranging not only over the integers but over all reals except the roots of p_2 . We show

that Algorithm 1 terminates after at most n_0 iterations. The previous formula implies

$$\begin{aligned} & \forall y_0, y_1 \forall n \geq n_0 : (y_0 \geq 0 \wedge y_1 \geq 0) \\ & \implies \frac{p_0(n)}{p_2(n)}y_0 + \frac{p_1(n)}{p_2(n)}y_1 \geq 0. \end{aligned}$$

Substituting $n \mapsto n - n_0$ leads to

$$\begin{aligned} & \forall y_0, y_1 \forall n \geq 0 : (y_0 \geq 0 \wedge y_1 \geq 0) \\ & \implies \frac{p_0(n+n_0)}{p_2(n+n_0)}y_0 + \frac{p_1(n+n_0)}{p_2(n+n_0)}y_1 \geq 0. \end{aligned}$$

As the variables y_0, y_1 range over all reals, the latter formula will remain true if we apply a substitution

$$\begin{aligned} y_0 & \mapsto r_0(n)y_0 + r_1(n)y_1, \\ y_1 & \mapsto r_2(n)y_0 + r_3(n)y_1, \end{aligned}$$

where $r_0, \dots, r_3$ are some rational functions in n . This gives

$$\begin{aligned} & \forall y_0, y_1 \forall n \geq 0 : (r_0(n)y_0 + r_1(n)y_1 \geq 0 \\ & \quad \wedge r_2(n)y_0 + r_3(n)y_1 \geq 0) \\ & \implies \frac{p_0(n+n_0)r_0(n) + p_1(n+n_0)r_2(n)}{p_2(n+n_0)}y_0 \\ & \quad + \frac{p_0(n+n_0)r_1(n) + p_1(n+n_0)r_3(n)}{p_2(n+n_0)}y_1 \geq 0. \end{aligned}$$

We are free to further modify this formula, without harming its truth, by imposing arbitrary additional conditions on the left hand side of the implication.

By choosing r_0, r_1, r_2, r_3 such that

$$\begin{aligned} f(n+n_0) &= r_0(n)f(n) + r_1(n)f(n+1) \\ f(n+n_0+1) &= r_2(n)f(n) + r_3(n)f(n+1) \end{aligned}$$

and adding constraints $q_{i,0}(n)y_0 + q_{i,1}(n)y_1 \geq 0$ encoding $f(n+i) \geq 0$ ($i = 0, \dots, n_0 - 1$) on the hypothesis part, we obtain precisely the formula $\Phi(n_0)$ as defined in Section 3.1 and used in Algorithm 1. As the formula is true, the algorithm terminates in the n_0 th iteration (or earlier), as we wanted to show. $\square$

REMARK 1. Algorithm 1 fails to terminate for positive u . To see this, consider the C-finite recurrence

$$f(n+2) - (u+1)f(n+1) + uf(n) = 0$$

for some $u \in (0, 1)$. If Algorithm 1 applied to this recurrence terminated in the n_0 th iteration, for some $n_0 \geq 0$, then the truth of $\Phi(n_0)$ implies that no solution $f: \mathbb{N} \rightarrow K$ of the recurrence can have n_0 consecutive nonnegative terms followed by a negative term. (So that, if n_0 consecutive terms are found nonnegative, all subsequent terms must be nonnegative as well.)

To see that no such n_0 can exist for the C-finite recurrence above, it is sufficient to construct for every $n_0 \geq 0$ a solution which contains a run of exactly n_0 nonnegative terms. The general solution of the recurrence is $c_0 + c_1u^n$ for some constants $c_0, c_1 \in K$. It is easily checked that the choice $c_0 = -1, c_1 = u^{-n_0+1}$ has the desired property.

The argument extends, at least for generic initial values, to P-finite balanced recurrence equations, using the fact that the recurrence admits two solutions $f_1, f_2: \mathbb{N} \rightarrow K$ with $f_1(n+1)/f_1(n) \xrightarrow{n \rightarrow \infty} 1$ and $f_2(n+1)/f_2(n) \xrightarrow{n \rightarrow \infty} u$.

THEOREM 3. *If $u \in (-1, 1) \setminus \{0\}$, then Algorithm 2 terminates for generic initial values.*

PROOF. Consider the set $D_3 \subseteq \mathbb{R}^3$ consisting of all points (c_0, c_1, μ) satisfying

$$0 < \mu < 1 \wedge \mu < c_1 < 2 \wedge \mu(\mu - c_1) < c_0 < 1$$

and the set

$$D_2 := \{ (c_0, c_1) \in \mathbb{R}^2 : 0 < c_1 < 2 \wedge -\frac{1}{4}c_1^2 < c_0 < 1 \}.$$

It can be shown by CAD computations that

$$\forall (c_0, c_1) \in D_2 \exists \mu \in (0, 1) : (c_0, c_1, \mu) \in D_3 \quad (1)$$

and that

$$\begin{aligned} \forall (c_0, c_1, \mu) \in D_3 \forall y_0, y_1 \in \mathbb{R} : \\ (y_0 \geq 0 \wedge y_1 \geq \mu y_0) \implies c_0 y_0 + c_1 y_1 \geq \mu y_1. \end{aligned} \quad (2)$$

Since $\frac{1}{4}(u+1)^2 > u$ for all $u \in (-1, 1)$, the set D_2 contains in particular the point $(-u, u+1)$ where u is from the statement of the theorem. Because of (1), there exists $\mu \in (0, 1)$ with $(-u, u+1, \mu) \in D_3$. Since D_3 is open, there exists $\varepsilon > 0$ such that

$$U := (-u - \varepsilon, -u + \varepsilon) \times (u + 1 - \varepsilon, u + 1 + \varepsilon) \times \{\mu\} \subseteq D_3.$$

Because of

$$\frac{p_0(n)}{p_2(n)} \xrightarrow{n \rightarrow \infty} -u \quad \text{and} \quad \frac{p_1(n)}{p_2(n)} \xrightarrow{n \rightarrow \infty} u + 1,$$

there exists $\xi \in \mathbb{N}$ such that

$$\left(\frac{p_0(n)}{p_2(n)}, \frac{p_1(n)}{p_2(n)}, \mu \right) \in U \subseteq D_3$$

for all $n \geq \xi$. Together with (2), this implies

$$\begin{aligned} \exists \mu \in (0, 1) \exists \xi \in \mathbb{N} \forall n \geq \xi \forall y_0, y_1 \in \mathbb{R} : \\ (y_0 \geq 0 \wedge y_1 \geq \mu y_0) \implies \frac{p_0(n)}{p_2(n)} y_0 + \frac{p_1(n)}{p_2(n)} y_1 \geq \mu y_1. \end{aligned}$$

Therefore, the set

$$C := \{ (\xi, \mu) \in (0, \infty) \times (0, 1) : \Phi(\xi, \mu) \}$$

with $\Phi(\xi, \mu)$ as used in Algorithm 2 is not empty.

Fix some point $(\xi, \mu) \in C$. Then it is immediate by the defining formula that also $(\xi', \mu) \in C$ for every $\xi' > \xi$, so

$$(\xi, \infty) \times \{\mu\} \subseteq C.$$

Let now $f: \mathbb{N} \rightarrow K$ be the sequence to which Algorithm 2 is applied. Then, because the eigenvalues of its defining recurrence are 1 and u and we have $|u| < 1$ and we assume generic initial values, we have

$$\frac{f(n+1)}{f(n)} \xrightarrow{n \rightarrow \infty} 1.$$

Since $\mu < 1$, this implies the existence of an index $m \in \mathbb{N}$ such that $f(n) \geq 0$ for all $n \geq m$ and $f(n+1)/f(n) \geq \mu$ for all $n \geq m$, so that we get $f(n+1) \geq \mu f(n)$ for all $n \geq m$.

It follows that the algorithm terminates no later than at iteration $\max(m, \xi)$. $\square$

REMARK 2. *The defining inequalities of D_3 in the preceding proof were found by quantifier elimination applied to formula (2) with the first quantifier dropped. This computation as well as the CAD computations referred to in the proof*

were performed with Mathematica's built-in implementation of CAD [18, 19]. The computation time is negligible for all of them and we are sure that other implementations [7, 3, etc.] would have no problem with them either.

EXAMPLE 4. *The restriction to generic initial values in Theorem 3 is essential: let $f: \mathbb{N} \rightarrow K$ be defined via*

$$\begin{aligned} (n+3)^2 f(n+2) - \frac{1}{2}(n+2)(3n+11)f(n+1) \\ + \frac{1}{2}(n+4)(n+1)f(n) = 0 \end{aligned}$$

and $f(0) = 1, f(1) = 1/4$. Then we have $f(n) = 2^{-n}/(n+1)$ for all $n \in \mathbb{N}$ and so in particular $f(n) \geq 0$ ($n \in \mathbb{N}$).

Algorithm 2 finds

$$\Phi(\xi, \mu) \equiv \frac{1}{2} \leq \mu \leq 1 \wedge \xi \geq \frac{17 - 12\mu - \sqrt{25 - 16\mu}}{2\mu - 3}$$

in Step 1 and continues by searching for an index n with $f(n+1) \geq \frac{1}{2}f(n)$. As no such index exist, the search continues forever.

The general solution of the defining recurrence for f is

$$c_0 + c_1 \frac{2^{-n}}{n+1}$$

and we will have $c_0 \neq 0$ for a generic choice of initial values. In these cases, the solution converges to c_0 and therefore eventually reaches an index n_0 with a term that is greater than half its predecessor.

4.3 Order Three

Consider now sequences $f: \mathbb{N} \rightarrow K$ defined by a balanced P-finite recurrence of third order,

$$p_3(n)f(n+3) - p_2(n)f(n+2) - p_1(n)f(n+1) - p_0(n)f(n) = 0.$$

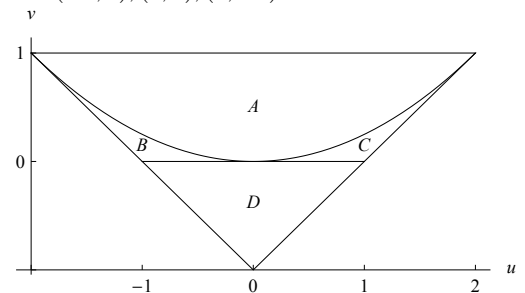
Again, we assume without loss of generality that 1 is a dominant eigenvalue and we let $u, v \in K$ be such that

$$(x-1)(x^2 + ux + v) = x^3 - (1-u)x^2 - (u-v)x - v$$

is the characteristic polynomial of the recurrence under consideration. The condition that the two roots of the quadratic factor belong to the interior of the complex unit disc translates into the condition

$$|u| - 1 < v < 1$$

for the coefficients of the polynomial. The points $(u, v) \in K^2$ satisfying this condition form the interior of the triangle with corners at $(-2, 1), (2, 1), (0, -1)$:



Just for the sake of orientation: the polynomial $x^2 + ux + v$ has two complex conjugate roots in region A, two positive real roots in region B, two negative real roots in region C, and a positive as well as a negative root in region D.

We want to identify regions of the triangle corresponding to recurrence equations on which Algorithms 1 and 2 terminate. Only for Algorithm 2 we have a satisfactory result, so let us consider this case first.

THEOREM 4. *If $|u| - 1 < v < 1$ and $4v < (u + 1)^2$ and $u < 1$, then Algorithm 2 terminates for generic initial values.*

PROOF. Consider the set $D_4 \subseteq \mathbb{R}^4$ consisting of all points (c_0, c_1, c_2, μ) satisfying

$$0 < \mu < 1 \wedge \mu < c_2 \wedge \mu(\mu - c_2) < c_1 \wedge \mu^3 - c_2\mu^2 - c_1\mu < c_0$$

and the set $D_3 \subseteq \mathbb{R}^3$ consisting of all points (c_0, c_1, c_2) with

$$\begin{aligned} & (0 < c_2 < 2 \wedge -\frac{1}{4}c_2^2 < c_1 < \min(3 - 2c_2, c_2^2) \\ & \wedge 2c_2^3 + 9c_1c_2 + 27c_0 + 2(c_2^2 + 3c_1)^{3/2} > 0) \\ & \vee (0 < c_2 < 1 \wedge c_1 \geq c_2^2 \wedge c_0 + c_1c_2 > 0). \end{aligned}$$

The following facts can be verified by CAD:

- $\forall (c_0, c_1, c_2) \in D_3 \exists \mu \in (0, 1) : (c_0, c_1, c_2, \mu) \in D_4$
- $\forall (c_0, c_1, c_2, \mu) \in D_4 \forall y_0, y_1, y_2 \in \mathbb{R} :$
 $(y_0 \geq 0 \wedge y_1 \geq \mu y_0 \wedge y_2 \geq \mu y_1)$
 $\implies c_0 y_0 + c_1 y_1 + c_2 y_2 \geq \mu y_2$
- $\forall u, v \in \mathbb{R} : (|u| - 1 < v < 1 \wedge 4v < (u + 1)^2 \wedge u < 1)$
 $\implies (v, u - v, 1 - u) \in D_3.$

Consequently, for u, v from the statement of the theorem there exists $\mu \in (0, 1)$ such that $(v, u - v, 1 - u, \mu) \in D_4$. Since D_4 is open, there exists $\varepsilon > 0$ such that

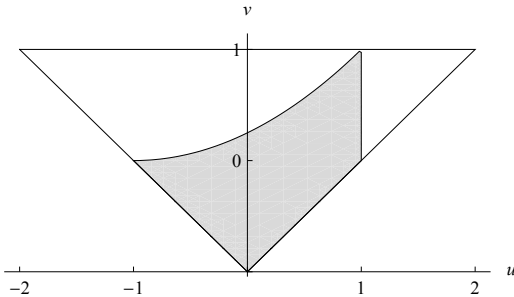
$$\begin{aligned} U := & (v - \varepsilon, v + \varepsilon) \times (u - v - \varepsilon, u - v + \varepsilon) \\ & \times (1 - u - \varepsilon, 1 - u + \varepsilon) \times \{\mu\} \subseteq D_4. \end{aligned}$$

Using

$$\frac{p_0(n)}{p_3(n)} \xrightarrow{n \rightarrow \infty} v, \quad \frac{p_1(n)}{p_3(n)} \xrightarrow{n \rightarrow \infty} u - v, \quad \frac{p_2(n)}{p_3(n)} \xrightarrow{n \rightarrow \infty} 1 - u,$$

the rest of the proof is fully analogous to the proof of Theorem 3. $\square$

The set of points (u, v) for which Theorem 4 asserts termination of Algorithm 2 is the shaded area in the figure below.



The truth of the formula

$$\begin{aligned} & \forall u \in (-1, 1) \forall v \in (|u| - 1, 1) : \\ & [\exists \mu > 0 \forall y_0, y_1, y_2 : \\ & (y_0 \geq 0 \wedge y_1 \geq \mu y_0 \wedge y_2 \geq \mu y_1) \\ & \implies v y_0 + (u - v) y_1 + (1 - u) y_2 \geq \mu y_2] \\ & \implies u < 1 \wedge 4v < (u + 1)^2 \end{aligned}$$

Copyright 2010 ACM 978-1-4503-0150-3/10/0007...\$10.00.

(as confirmed, once again, by a CAD computation) asserts that Theorem 4 is sharp.

We are not able to provide a sharp result for the terminating region of Algorithm 1. If we proceed to reason as in the proof of Theorem 2, we obtain termination for (u, v) restricted to the (open) triangle with vertices $(0, 0)$, $(1, 0)$, $(1, 1)$, essentially because of

$$\begin{aligned} & \forall u \in (0, 1) \forall v \in (0, u) \forall y_0, y_1, y_2 : \\ & (y_0 \geq 0 \wedge y_1 \geq 0 \wedge y_2 \geq 0) \\ & \implies v y_0 + (u - v) y_1 + (1 - u) y_2 \geq 0 \end{aligned}$$

and the convergences

$$\frac{p_0(n)}{p_3(n)} \xrightarrow{n \rightarrow \infty} v, \quad \frac{p_1(n)}{p_3(n)} \xrightarrow{n \rightarrow \infty} u - v, \quad \frac{p_2(n)}{p_3(n)} \xrightarrow{n \rightarrow \infty} 1 - u.$$

But this is not the entire terminating region. A larger portion of the terminating region can be identified by starting out with a formula corresponding to an induction hypothesis of length four. As the formula

$$\begin{aligned} & \forall y_0, y_1, y_2 : (y_0 \geq 0 \wedge y_1 \geq 0 \wedge y_2 \geq 0 \\ & \wedge v y_0 + (u - v) y_1 + (1 - u) y_2 \geq 0) \\ & \implies (1 - u) v y_0 + u(1 - u + v) y_1 + (1 - u + u^2 - v) y_2 \geq 0 \end{aligned}$$

is true for all (u, v) with

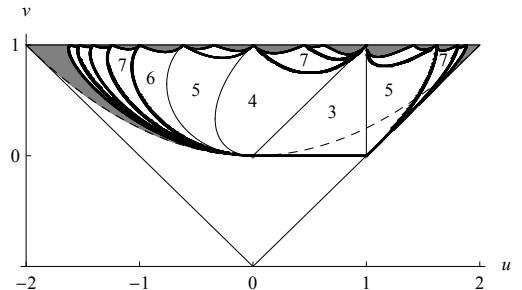
$$\begin{aligned} & u < 1 \wedge v > 0 \wedge 1 - u + u^2 - v > 0 \\ & \wedge (u > 0 \vee u^2 - v - uv + v^2 < 0), \end{aligned}$$

and as we have

$$\begin{aligned} & \frac{p_0(n)p_2(n+1)}{p_3(n)p_3(n+1)} \xrightarrow{n \rightarrow \infty} (1 - u)v, \\ & \frac{p_1(n)p_2(n+1) + p_0(n+1)p_3(n)}{p_3(n)p_3(n+1)} \xrightarrow{n \rightarrow \infty} u(1 - u + v), \\ & \frac{p_2(n)p_2(n+1) + p_1(n+1)p_3(n)}{p_3(n)p_3(n+1)} \xrightarrow{n \rightarrow \infty} 1 - u + u^2 - v, \end{aligned}$$

Algorithm 1 also terminates for all (u, v) satisfying the conditions stated above.

Starting out with a formula corresponding to a hypothesis of length five leads to a portion of the termination region whose description can be computed in a reasonable amount of time, but which is already too big to be reproduced here. For longer induction hypotheses, the computational effort for doing quantifier elimination becomes prohibitive. But it is still possible to determine experimentally the regions obtained by taking a particular length ϱ of the induction hypothesis taken as the starting point of the termination proof. The empiric results for induction hypotheses of length up to 10 are as follows (the numbers indicate the length of the induction hypothesis):



The picture suggests the following characterization for the full region of termination.

CONJECTURE 1. If $|u| - 1 < v < 1$ and

$$(u > 1 \wedge v > 0) \vee 4v > (u + 1)^2,$$

then Algorithm 1 terminates.

The conjecture is equivalent to saying that Algorithm 1 terminates if $x^2 + ux + v$ has no positive root. If the conjecture is true, then about 96.35% of the area of the triangle are covered by one of the two algorithms we considered.

5. REFERENCES

- [1] Horst Alzer, Stefan Gerhold, Manuel Kauers, and Alexandru Lupas. On Turán’s inequality for Legendre polynomials. *Expositiones Mathematicae*, 25(2):181–186, 2007.
- [2] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer, 2nd edition, 2006.
- [3] Chris W. Brown. QEPCAD B – a program for computing with semi-algebraic sets. *Sigsum Bulletin*, 37(4):97–108, 2003.
- [4] Bob F. Caviness and Jeremy R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation. Springer, 1998.
- [5] George E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. *Lecture Notes in Computer Science*, 33:134–183, 1975.
- [6] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Comput.*, 12(3):299–328, 1991.
- [7] Andreas Dolzmann and Thomas Sturm. Guarded expressions in practice. In *Proceedings of ISSAC’97*, 1997.
- [8] Graham Everest, Alf van der Poorten, Igor Shparlinski, and Thomas Ward. *Recurrence Sequences*, volume 104 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2003.
- [9] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.
- [10] Stefan Gerhold. *Combinatorial Sequences: Non-Holonomicity and Inequalities*. PhD thesis, RISC-Linz, Johannes Kepler Universität Linz, 2005.
- [11] Stefan Gerhold and Manuel Kauers. A procedure for proving special function inequalities involving a discrete parameter. In Manuel Kauers, editor, *Proceedings of ISSAC’05*, pages 156–162, 2005.
- [12] Stefan Gerhold and Manuel Kauers. A computer proof of Turán’s inequality. *Journal of Inequalities in Pure and Applied Mathematics*, 7(2):#42, 2006.
- [13] Manuel Kauers. Computer algebra and power series with positive coefficients. In *Proceedings of FPSAC’07*, 2007.
- [14] Manuel Kauers. Computer algebra and special function inequalities. In Tewodros Amdeberhan and Victor H. Moll, editors, *Tapas in Experimental Mathematics*, volume 457 of *Contemporary Mathematics*, pages 215–235. AMS, 2008.
- [15] Manuel Kauers and Peter Paule. A computer proof of Moll’s log-concavity conjecture. *Proceedings of the AMS*, 135(12):3847–3856, 2007.
- [16] Marc Mezzarobba and Bruno Salvy. Effective bounds for P-recursive sequences. *preprint*, ArXiv:0904.2452, 2009.
- [17] Veronika Pillwein. Positivity of certain sums over Jacobi kernel polynomials. *Advances in Applied Mathematics*, 41(3):365–377, 2008.
- [18] Adam Strzeboński. Solving systems of strict polynomial inequalities. *Journal of Symbolic Computation*, 29:471–480, 2000.
- [19] Adam Strzeboński. Cylindrical algebraic decomposition using validated numerics. *Journal of Symbolic Computation*, 41(9):1021–1038, 2006.
- [20] Jet Wimp and Doron Zeilberger. Resurrecting the asymptotics of linear recurrences. *Journal of Mathematical Analysis and Applications*, 111:162–176, 1985.

Complexity of Creative Telescoping for Bivariate Rational Functions*

Alin Bostan, Shaoshi Chen, Frédéric Chyzak
Algorithms Project-Team, INRIA
Paris-Rocquencourt
78153 Le Chesnay (France)
{alin.bostan,shaoshi.chen,frederic.chyzak}@inria.fr

Ziming Li
Key Laboratory of Mathematics Mechanization,
Academy of Mathematics and System Sciences
100190 Beijing (China)
zmli@mmrc.iss.ac.cn

ABSTRACT

The long-term goal initiated in this work is to obtain fast algorithms and implementations for definite integration in Almkvist and Zeilberger’s framework of (differential) creative telescoping. Our complexity-driven approach is to obtain tight degree bounds on the various expressions involved in the method. To make the problem more tractable, we restrict to *bivariate rational* functions. By considering this constrained class of inputs, we are able to blend the general method of creative telescoping with the well-known Hermite reduction. We then use our new method to compute diagonals of rational power series arising from combinatorics.

Categories and Subject Descriptors

I.1.2 [Computing Methodologies]: Symbolic and Algebraic Manipulations—*Algebraic Algorithms*

General Terms

Algorithms, Theory

Keywords

Hermite reduction, creative telescoping.

1. INTRODUCTION

The long-term goal of the research initiated in the present work is to obtain fast algorithms and implementations for the definite integration of general special functions, in a complexity-driven perspective.

As most special-function integrals cannot be expressed in closed form, their evaluation cannot be based on table look-ups only, and even when closed forms are available, they may prove to be intractable in further manipulations. In both cases, the difficulty can be mitigated by representing functions by annihilating differential operators. This motivated Zeilberger to introduce a method now known as *cre-*

ative telescoping [18], which applies to a large class of special functions: the D-finite functions [14] defined by sets of linear differential equations of any order, with polynomial coefficients. Zeilberger’s method applies in general to multiple integrals and sums.

A sketch of Zeilberger’s method is as follows. Given a D-finite function f of the variables x and y , the definite integral $F(x) = \int_{\alpha}^{\beta} f(x, y) dy$ is D-finite, and a linear differential equation satisfied by F can be constructed [18]. To explain this, let k be a field of characteristic zero, D_x and D_y be the usual derivations on the rational-function field $k(x, y)$, both restricting to zero on k , and let $k(x, y)\langle D_x, D_y \rangle$ be the ring of linear differential operators over $k(x, y)$. The heart of the method is to solve the *differential telescoping equation* (1) below for $L \in k[x]\langle D_x \rangle \setminus \{0\}$ and $g = R(f)$ for some $R \in k(x, y)\langle D_x, D_y \rangle$. The operator L is called a *telescoper* for f , and g a *certificate* of L for f . Under the assumption

$$\lim_{y \rightarrow \alpha} g(x, y) = \lim_{y \rightarrow \beta} g(x, y) \quad \text{for } x \text{ in some domain,}$$

$L(x, D_x)$ is then proved to be an annihilator of F .

The main emphasis in works since the 1990’s has been on finding telescopers of order minimal over all telescopers for f , which are called *minimal telescopers*. (Two minimal telescopers differ by a multiplicative factor in $k(x)$.) In view of the computational difficulty of solving (1), there has been special attention to subclasses of inputs. Of particular importance is the case of hyperexponential functions, defined by first-order differential equations, studied by Almkvist and Zeilberger in [1]. Their method is a direct differential analogue of Zeilberger’s algorithm for the recurrence case [19].

On the other hand, very little is known about the complexity of creative telescoping: the only related result seems to be an analysis in [9] of an algorithm for hyperexponential indefinite integration. In order to get complexity estimates, we simplify the problem by restricting to a smaller class of inputs, namely that of bivariate rational functions. Although restricted, this class already has many applications, for instance in combinatorics, where many nontrivial problems are encoded as diagonals of rational formal power series, themselves expressible as integrals. Our goal thus reads as follows.

Problem Given $f = P/Q \in k(x, y) \setminus \{0\}$, find a pair (L, g) with $L = \sum_{i=0}^p \eta_i(x) D_x^i$ in $k[x]\langle D_x \rangle \setminus \{0\}$ and g in $k(x, y)$ such that

$$L(x, D_x)(f) = D_y(g). \quad (1)$$

By considering this more constrained class of inputs, we are indeed able to blend the general method of creative telescoping with the well-known Hermite reduction [10].

*We warmly thank the referees for their very helpful comments. — AB and FC were supported in part by the Microsoft Research – Inria Joint Centre, and SC and ZL by a grant of the National Natural Science Foundation of China (No. 60821002).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

	Method	$\deg_{D_x}(L)$	$\deg_x(L)$	$\deg_x(g)$	$\deg_y(g)$	Complexity
Minimal	Hermite reduction (new)	$\leq d_y$	$\mathcal{O}(d_x d_y^2)$	$\mathcal{O}(d_x d_y^2)$	$\mathcal{O}(d_y^2)$	$\tilde{\mathcal{O}}(d_x d_y^{\omega+3})$ Las Vegas
Telescoper	Almkvist and Zeilberger	$\leq d_y$	$\mathcal{O}(d_x d_y^2)$	$\mathcal{O}(d_x d_y^2)$	$\mathcal{O}(d_y^2)$	$\tilde{\mathcal{O}}(d_x d_y^{2\omega+2})$ Las Vegas
Nonminimal	Lipshitz elimination	$\leq 6(d_x + 1)(d_y + 1)$	$\mathcal{O}(d_x d_y)$	$\mathcal{O}(d_x^2 d_y)$	$\mathcal{O}(d_x d_y^2)$	$\mathcal{O}(d_x^{3\omega} d_y^{3\omega})$ deterministic
Telescoper	Cubic size	$\leq 6d_y$	$\mathcal{O}(d_x d_y)$	$\mathcal{O}(d_x d_y)$	$\mathcal{O}(d_y^2)$	$\mathcal{O}(d_x^\omega d_y^{3\omega})$ deterministic

Figure 1: Complexity of creative telescoping methods (under Hyp. (H')), together with bounds on output

Essentially two algorithms for minimal telescopers can be found in the literature: The classical way [1] is to apply a differential analogue of Gosper's indefinite summation algorithm, which reduces the problem to solving an auxiliary linear differential equation for polynomial solutions. An algorithm developed later in [7] (see also [12]) performs Hermite reduction on f to get an additive decomposition of the form $f = D_y(a) + \sum_{i=1}^n u_i/v_i$, where the u_i and v_i are in $k(x)[y]$ and the v_i are squarefree. Then, the algorithm in [1] is applied to each u_i/v_i to get a telescoper L_i minimal for it. The least common left multiple of the L_i 's is then proved to be a minimal telescoper for f . This algorithm performs well only for specific inputs (both in practice and from the complexity viewpoint), but it inspired our Lemma 22 via [12].

As a first contribution in this article, we present a new, provably faster algorithm for computing minimal telescopers for bivariate rational functions. Instead of a single use of Hermite reduction as in [12], we apply Hermite reduction to the $D_x^i(f)$'s, iteratively for $i = 0, 1, \dots$, which yields

$$D_x^i(f) = D_y(g_i) + \frac{w_i}{w} \quad (2)$$

for some factor w of the squarefree part of the denominator of f . If $\eta_0, \dots, \eta_\rho \in k(x)$ are not all zero and such that $\sum_{i=0}^\rho \eta_i w_i = 0$, then the operator $\sum_{i=0}^\rho \eta_i D_x^i$ is a telescoper for f , and more specifically, the first nontrivial linear relation obtained in this way yields a minimal telescoper for f .

As a second contribution, we give the first proof of a polynomial complexity for creative telescoping on a specific class of inputs, namely on bivariate rational functions. For *minimal* telescopers, only a polynomial bound on d_x (but none on d_y) was given for special inputs in [7]; more specifically, we derive complexity estimates for all mentioned methods (see Fig. 1), showing that our approach is faster. Furthermore, we analyse the bidegrees of *non minimal* telescopers generated by other approaches: Lipshitz' work [13] can be rephrased into an existence theorem for telescopers with polynomial size; the approach followed in the recent work on algebraic functions [3] leads to telescopers of smaller degree sizes. These are new instances of the philosophy, promoted in [3], that relaxing minimality can produce smaller outputs.

A third contribution is a fast Maple implementation [20], incorporating a careful implementation of the original Hermite reduction algorithm, making use of the special form of w_i/w in (2) and of usual modular techniques (probabilistic rank estimate) to determine when to invoke the solver for linear algebraic equations. Experimental results indicate that our implementation outperforms Maple's core routine.

Note that for the fastest method we propose, denoted by H1 in Tables 1–3, we chose to output the certificate as a mere sum of (small) rational functions, without any form of normalisation. This choice seems to be uncommon for creative-telescoping algorithms, but a motivation is how the certificate is used in practice: Very often, like for applications to diagonals in §5, the certificate is actually not needed. In

other applications, the next step of the method of creative telescoping is to integrate (1) between α and β , leading to $L(F)(x) = g(x, \alpha) - g(x, \beta)$. Therefore, only evaluations of the certificate are really needed, and normalisation can be postponed to after specialising at α and β .

The end of this section, §1.1, provides classical complexity results, notation, and hypotheses that will be used throughout. We then study Hermite reduction over $k(x)$ in §2, proving output degree bounds and a low-complexity algorithm. This is then applied in §3 to derive our new algorithm for creative telescoping, and to compare its complexity with that of Almkvist and Zeilberger's approach. For nonminimal telescopers, we show the existence of some of lower arithmetic size in §4: cubic for nonminimal order instead of quartic for minimal order. See the summary in Figure 1, where the low complexity of algorithms for minimal telescopers relies on Storjohann and Villard's algorithms [17], thus inducing a *certified* probabilistic feature. We apply our results to the calculation of diagonals in §5, and describe our implementation and comment on execution timings in §5.

1.1 Background on complexity — Notation

We recall basic notation and complexity facts for later use. Let k be again a field of characteristic zero. Unless otherwise specified, all complexity estimates are given in terms of arithmetical operations in k , which we denote by “ops”. Let $k[x]_{\leq d}^{m \times n}$ be the set of $m \times n$ matrices with coefficients in $k[x]$ of degree at most d . Let $\omega \in [2, 3]$ be a feasible exponent of matrix multiplication, so that two matrices from $k^{n \times n}$ can be multiplied using $\mathcal{O}(n^\omega)$ ops. Facts 1 and 2 below show the complexity of multipoint evaluation, rational interpolation, and algebraic operations on polynomial matrices using fast arithmetic, where the notation $\tilde{\mathcal{O}}(\cdot)$ indicates cost estimates with hidden logarithmic factors [6, Def. 25.8].

Fact 1 For $p \in k[x]$ of degree less than n , pairwise distinct $u_0, \dots, u_{n-1}$ in k , and $v_0, \dots, v_{n-1} \in k$, we have:

- (i) Evaluating p at the u_i 's takes $\tilde{\mathcal{O}}(n)$ ops.
- (ii) For $m \in \{1, \dots, n\}$, constructing $f = s/t \in k(x)$ with $\deg_x(s) < m$ and $\deg_x(t) \leq n - m$ such that $t(u_i) \neq 0$ and $f(u_i) = v_i$ for $0 \leq i \leq n - 1$ takes $\tilde{\mathcal{O}}(n)$ ops.

Fact 2 For M in $k[x]_{\leq d}^{m \times n}$, $d > 0$, we have:

- (i) If $M = \begin{pmatrix} M_1 & M_2 \end{pmatrix}$ is an invertible $n \times n$ matrix with $M_i \in k[x]_{\leq d_i}^{n \times n_i}$, where $i = 1, 2$ and $n_1 + n_2 = n$, then the degree of $\det(M)$ is at most $n_1 d_1 + n_2 d_2$.
- (ii) If $M = \begin{pmatrix} M_1 & M_2 \end{pmatrix}$ is not of full rank and with $M_i \in k[x]_{\leq d_i}^{m \times n_i}$, where $i = 1, 2$ and $n_1 + n_2 = n$, then there exists a nonzero $u \in k[x]^n$ with coefficients of degree at most $n_1 d_1 + n_2 d_2$ such that $Mu = 0$.
- (iii) The rank r and a basis of the null space of M can be computed using $\tilde{\mathcal{O}}(nmr^{\omega-2}d)$ ops.

(For proofs, see [6, Cor. 10.8, 5.18, 11.6] and [17, Th. 7.3].)

We call *squarefree factorisation* of $Q \in k[x, y] \setminus k[x]$ w.r.t. y the unique product $qQ_1Q_2^2 \cdots Q_m^m$ equal to Q for $q \in k[x]$ and $Q_i \in k[x, y]$ satisfying $\deg_y(Q_m) > 0$ and such that the Q_i 's are primitive, squarefree, and pairwise coprime. The *squarefree part* Q^* of Q w.r.t. y is the product $Q_1Q_2 \cdots Q_m$. Let Q^- denote the polynomial Q/Q^* , and $\text{lc}_y(Q)$ the leading coefficient of Q w.r.t. y . The following two formulas about Q , Q^* , and Q^- can be proved by mere calculations.

Fact 3 *Let $\hat{Q}_i$ denote Q^*/Q_i . Then we have*

- (i) $Q^*D_y(Q^-)/Q^- = \sum_{i=1}^m (i-1)\hat{Q}_i D_y(Q_i) \in k[x, y];$
- (ii) $D_y(Q)/Q^- = \sum_{i=1}^m i\hat{Q}_i D_y(Q_i) \in k[x, y].$

Let $f = P/Q$ be a nonzero element in $k(x, y)$, where P, Q are two coprime polynomials in $k[x, y]$. The degree of f in x is defined to be $\max\{\deg_x(P), \deg_x(Q)\}$, and denoted by $\deg_x(f)$. The degree of f in y is defined similarly. The *bidegree* of f is the pair $(\deg_x(f), \deg_y(f))$, which is denoted by $\text{bideg}(f)$. The bidegree of f is said to be *bounded (above)* by (α, β) , written $\text{bideg}(f) \leq (\alpha, \beta)$, when $\deg_x(f) \leq \alpha$ and $\deg_y(f) \leq \beta$.

We say that $f = P/Q$ is *proper* if the degree of P in y is less than that of Q . For creative telescoping, we may always assume w.l.o.g. that $f = P/Q$ is proper. If not, rewrite $f = D_y(p) + \bar{f}$ with $p \in k(x)[y]$ and $\bar{f}$ proper. A telescoper L for f with certificate $\bar{g}$ is a telescoper for f with certificate $L(p) + \bar{g}$.

Hypothesis (H) *From now on, P and Q are assumed to be nonzero polynomials in $k[x, y]$ such that $\deg_y(P) < \deg_y(Q)$, $\gcd(P, Q) = 1$, and Q is primitive w.r.t. y .*

Notation *From now on, we write (d_x, d_y) , (d_x^*, d_y^*) , and (d_x^-, d_y^-) for the bidegrees of Q , Q^* , and Q^- , respectively.*

The following hypothesis makes our estimates concise.

Hypothesis (H') *Occasionally, we shall require the extended hypothesis: Hypothesis (H) and $\deg_x(P) \leq d_x$.*

2. HERMITE REDUCTION

Let K be a field of characteristic zero, either k or $k(x)$ in what follows. Let $K(y)$ be the field of rational functions in y over K , and D_y be the usual derivation on it. For a rational function $f \in K(y)$, *Hermite reduction* [10] computes rational functions g and $r = a/b$ in $K(y)$ satisfying

$$f = D_y(g) + r, \quad \deg_y(a) < \deg_y(b), \quad b \text{ is squarefree.} \quad (3)$$

Horowitz and Ostrogradsky's method [15, 11] computes the same decomposition as in (3) by solving a linear system. For the details of those methods, see [4, Chapter 2].

Lemma 4 *If f is proper, a pair (g, r) satisfying (3) for proper g, r is unique.*

PROOF. This is a consequence of [11, Theorem 2.10] after writing r as a sum $\sum_{i=1}^m \alpha_i/(x - b_i)$ and integrating. $\square$

Lemma 5 *Let f be a nonzero rational function in $K(y)$ of degree at most n in y , then Hermite reduction on f can be performed using $\tilde{O}(n)$ operations in K .*

PROOF. See [6, Theorem 22.7]. $\square$

In contrast, the method of Horowitz and Ostrogradsky takes $\mathcal{O}(n^\omega)$ operations in K [6, § 22.2]. Thus, Hermite's method is quasi-optimal and asymptotically faster than the former.

From now on, we fix $K = k(x)$ and analyse the complexity of Hermite reduction over $k(x)$ in terms of operations in k . To this end, we use an evaluation-interpolation approach.

2.1 Output size estimates

We derive an upper bound on the bidegrees of g and r satisfying (3) by studying the linear system in [11].

Analysing Hermite reduction (under (H)) shows the existence of $A, a \in k(x)[y]$ with $\deg_y(A) < d_y^-$, $\deg_y(a) < d_y^*$ and

$$\frac{P}{Q} = D_y \left(\frac{A}{Q^-} \right) + \frac{a}{Q^*}. \quad (4)$$

In order to bound the bidegrees of A and a , we reformulate (4) into the equivalent form

$$P = Q^* D_y(A) - \left(\frac{Q^* D_y(Q^-)}{Q^-} \right) A + Q^- a, \quad (5)$$

where $Q^* D_y(Q^-)/Q^-$ is a polynomial in $k[x, y]$ of bidegree at most $(d_x^*, d_y^* - 1)$ by Fact 3. Viewing A and a as polynomials in $k(x)[y]$ with undetermined coefficients, we form the following linear system, equivalent to (5),

$$(\mathcal{H}_1 \ \mathcal{H}_2) \begin{pmatrix} \hat{A} \\ \hat{a} \end{pmatrix} = \hat{P}, \quad (6)$$

where $\mathcal{H}_1 \in k[x]_{\leq d_x^*}^{d_y \times d_y^-}$, $\mathcal{H}_2 \in k[x]_{\leq d_x^-}^{d_y \times d_y^*}$, and $\hat{A}$, $\hat{a}$, and $\hat{P}$ are the coefficient vectors of A , a , and P with sizes d_y^- , d_y^* , and d_y , respectively. Under the constraint of properness of A/Q^- and a/Q^* , (A, a) is unique by Lemma 4. Then (6) has a unique solution, which leads to the following lemma.

Lemma 6 *The matrix $(\mathcal{H}_1 \ \mathcal{H}_2)$ is invertible over $k(x)$.*

As the matrix $(\mathcal{H}_1 \ \mathcal{H}_2)$ is uniquely defined by Q , we call it the matrix *associated* with Q , denoted by $\mathcal{H}(Q)$. Let δ be its determinant, so that $\deg_x(\delta) \leq \mu := d_x^* d_y^- + d_x^- d_y^*$ by Fact 2(i). For later use, we also define δ' as the determinant of $\mathcal{H}(Q^{*2})$, so that $\deg_x(\delta') \leq \mu' := 2d_x^* d_y^*$ by Fact 2(i) and since $(Q^{*2})^- = Q^*$.

Lemma 7 *There exist $B, b \in k[x, y]$ with $\deg_y(B) < d_y^-$ and $\deg_y(b) < d_y^*$, and such that:*

- (i) $\frac{P}{Q} = D_y \left(\frac{B}{\delta Q^-} \right) + \frac{b}{\delta' Q^*};$
- (ii) $\deg_x(B) \leq \mu - d_x^* + \deg_x(P)$ and $\deg_x(b) \leq \mu - d_x^- + \deg_x(P).$

PROOF. Applying Cramer's rule to (6) leads to (i). Assertion (ii) next follows by determinant expansions. $\square$

In what follows, we shall encounter proper rational functions with denominator Q satisfying $Q = Q^{*2}$. The following lemma is an easy corollary of Lemma 7 for such functions.

Corollary 8 *Assuming $Q = Q^{*2}$ in addition to Hypothesis (H), there exist $B, b \in k[x, y]$ with $\deg_y(B)$ and $\deg_y(b)$ less than d_y^* , and such that*

- (i) $\frac{P}{Q^{*2}} = D_y \left(\frac{B}{\delta' Q^*} \right) + \frac{b}{\delta' Q^*};$
- (ii) $\deg_x(B)$ and $\deg_x(b)$ are bounded by $\mu' - d_x^* + \deg_x(P).$

2.2 Algorithm by evaluation and interpolation

We observe that an asymptotically optimal complexity can be achieved by evaluation and interpolation at each step of Hermite reduction over $k(x)$. This inspires us to adapt Gerhard's modular method [8, 9] to $k(x, y)$. Recall that, by Hyp. (H), $Q \in k[x, y]$ is nonzero and primitive over $k[x]$.

Definition An element $x_0 \in k$ is lucky if $\text{lc}_y(Q)(x_0) \neq 0$ and $\deg_y(\gcd(Q(x_0, y), D_y(Q(x_0, y)))) = d_y^-$.

Lemma 9 There are at most $d_x(2d_y^* - 1)$ unlucky points.

PROOF. Let $\sigma \in k[x]$ be the d_y^- -th subresultant w.r.t. y of Q and $D_y(Q)$. By [9, Corollary 5.5], all unlucky points are in the set $U = \{x_0 \in k \mid \sigma(x_0) = 0\}$. By [9, Corollary 3.2(ii)], $\deg_x(\sigma) \leq d_x(2d_y^* - 1)$. $\square$

Lemma 10 Let B, b , and δ be the same as in Lemma 7, and let $x_0 \in k$ be lucky. Then $\delta(x_0) \neq 0$ and $(B(x_0, y), b(x_0, y))$ is the unique pair such that

$$\frac{P(x_0, y)}{Q(x_0, y)} = D_y \left(\frac{B(x_0, y)}{\delta(x_0)Q^-(x_0, y)} \right) + \frac{b(x_0, y)}{\delta(x_0)Q^*(x_0, y)}. \quad (7)$$

PROOF. By the luckiness of x_0 , $\deg_y(Q(x_0, y)) = d_y$ and $Q(x_0, y)^- = Q^-(x_0, y)$, so $Q(x_0, y)^* = Q^*(x_0, y)$. This implies $\mathcal{H}(Q)(x_0, y) = \mathcal{H}(Q(x_0, y))$, which, by Lemma 6, is invertible over $k(x)$. Hence $\delta(x_0) \neq 0$, and the evaluation at $x = x_0$ of the equality in Lemma 7(i) is well-defined. Thus, $(B(x_0, y), b(x_0, y))$ is a solution of (7). Uniqueness follows from Lemma 4. $\square$

Theorem 11 Algorithm *HermiteEvalInterp* in Figure 2 is correct and takes $\tilde{O}(d_x d_y^2 + \deg_x(P) d_y)$ ops.

PROOF. Set ν to $d_x(2d_y^* - 1)$. Lemma 9 implies that the $\lambda + 1$ lucky points found in Step 3 are all less than $\lambda + \nu + 1$. By Lemmas 4 and 7(i), $A = B/\delta$ and $a = b/\delta$. By Lemma 10, $A_0 = B(x_0, y)/\delta(x_0)$ and $a_0 = b(x_0, y)/\delta(x_0)$. By Lemma 7(ii) and since $\deg_x(\delta) \leq \mu$, it suffices to rationally interpolate A and a from values at $\lambda + 1$ lucky points. This shows the correctness. The dominant computation in Step 1 is the gcd, which takes $\tilde{O}(d_x d_y)$ ops by [6, Cor. 11.9]. For each integer $i \leq \lambda + \nu$, testing luckiness amounts to evaluations at x_0 and computing $\gcd(Q(x_0, y), D_y(Q(x_0, y)))$, which takes $\tilde{O}(d_y)$ ops by Fact 1(i) and [6, Cor. 11.6]. Then, generating S in Step 3 costs $\tilde{O}((\lambda + \nu + 1)d_y)$ ops. By Fact 1(i), evaluations in Step 4 take $\tilde{O}((\lambda + 1)d_y)$ ops. For each $x_0 \in S$, the cost of the Hermite reduction in Step 4 is $\tilde{O}(d_y)$ ops by Lemma 5. Thus, the total cost of Step 4 is $\tilde{O}((\lambda + 1)d_y)$ ops. By Fact 1(ii), Step 5 takes $\tilde{O}((\lambda + 1)d_y)$ ops. Since $\lambda \leq 2d_x d_y + \deg_x(P)$ and $\nu \leq 2d_x d_y$, the total cost is as announced. $\square$

As the generic output size of Hermite reduction is proportional to λd_y , which is $\mathcal{O}((d_x d_y + \deg_x(P))d_y)$, Algorithm *HermiteEvalInterp* has quasi-optimal complexity.

3. MINIMAL TELESCOPERS

We analyse two algorithms for constructing minimal telescopers for bivariate rational functions and their certificates.

Algorithm *HermiteEvalInterp*(P, Q)

INPUT: $P, Q \in k[x, y]$ satisfying Hypothesis (H).

OUTPUT: $(A, a) \in k(x)[y]^2$ solving (4).

1. Compute $Q^- := \gcd(Q, D_y(Q))$ and $Q^* := Q/Q^-$;
2. Set $\lambda := 2(d_x^* d_y^- + d_y^* d_x^-) + \deg_x(P) - \min\{d_x^-, d_x^*\}$;
3. Set S to the set of $\lambda + 1$ smallest nonnegative integers that are lucky for Q ;
4. For each $x_0 \in S$, compute $(A_0, a_0) \in k[y]^2$ such that

$$\frac{P(x_0, y)}{Q(x_0, y)} = D_y \left(\frac{A_0}{Q^-(x_0, y)} \right) + \frac{a_0}{Q^*(x_0, y)}$$

using Hermite reduction over k ;

5. Compute $(A, a) \in k(x)[y]$ by rational interpolation and return this pair.

Figure 2: Hermite reduction over $k(x)$ via evaluation and interpolation.

3.1 Hermite reduction approach

We design a new algorithm, presented in Figure 3, to compute minimal telescopers for rational functions by basing on Hermite reduction. For $f = P/Q \in k(x, y)$ and $i \in \mathbb{N}$, Hermite reduction decomposes $D_x^i(f)$ into

$$D_x^i(f) = D_y(g_i) + r_i, \quad (8)$$

where $g_i, r_i \in k(x, y)$ are proper. Since the squarefree part of the denominator of $D_x^i(f)$ divides Q^* , so does the denominator of r_i . The following lemma shows that (8) recombines into telescopers and certificates; next, Lemma 13 implies that the first pair obtained in this way by Algorithm *HermiteTelescoping* in Figure 3 yields a minimal telescoper.

Lemma 12 The rational functions $r_0, \dots, r_{d_y^*}$ are linearly dependent over $k(x)$.

PROOF. The constraints on r_i imply $\deg_y(r_i Q^*) < d_y^*$ for all $i \in \mathbb{N}$, from which follows the existence of a nontrivial linear dependence among the r_i 's over $k(x)$. $\square$

Lemma 13 An integer ρ is minimal such that $\sum_{i=0}^{\rho} \eta_i r_i = 0$ for $\eta_0, \dots, \eta_{\rho} \in k(x)$ not all zero if and only if $\sum_{i=0}^{\rho} \eta_i D_x^i$ is a minimal telescoper for f with certificate $\sum_{i=0}^{\rho} \eta_i g_i$.

PROOF. Multiplying (8) by η_i before summing yields

$$L(f) = D_y \left(\sum_{i=0}^{\rho} \eta_i g_i \right) + \sum_{i=0}^{\rho} \eta_i r_i \quad \text{for} \quad L := \sum_{i=0}^{\rho} \eta_i D_x^i,$$

where the first two sums are proper. Thus, by Lemma 4, L is a telescoper of order ρ for f with certificate $\sum_{i=0}^{\rho} \eta_i g_i$ if and only if $\sum_{i=0}^{\rho} \eta_i r_i = 0$ with $\eta_{\rho} \neq 0$. The lemma follows. $\square$

3.1.1 Order bounds for minimal telescopers

Lemmas 12 and 13 combine into an upper bound on the order of minimal telescopers for f .

Corollary 14 Minimal telescopers have order at most d_y^* .

The bound $6d_y$ is shown in [3] for rational functions of the form $yD_y(Q)/Q$ with $Q \in k[x, y]$. Apagodu and Zeilberger [2]

obtain a similar bound for a class of nonrational hyperexponential functions, but their proof does not seem to apply to rational functions, as it heavily relies on the presence of a nontrivial exponential part.

We also derive a lower bound on the order of the minimal telescoper, to be used as an optimisation at the end of § 3.1.3: choosing a lucky $x_0 \in k$, next applying Hermite reduction in $k(y)$ to $D_x^i(f)(x_0, y)$, yields

$$D_x^i(f)(x_0, y) = D_y(g_{0,i}) + r_{0,i}, \quad (9)$$

where $g_{0,i}, r_{0,i} \in k(y)$ are proper and the denominator of $r_{0,i}$ divides $Q^*(x_0, y)$. Let ρ_0 be the smallest integer such that $r_{0,0}, \dots, r_{0,\rho_0}$ are linearly dependent over k .

Lemma 15 *A minimal telescoper has order at least ρ_0 .*

PROOF. We first claim that $r_{0,i} = r_i(x_0, y)$, for r_i as in (8). Note that the squarefree part w.r.t. y of the denominator of $D_x^i(f)$ divides Q^* for all $i \in \mathbb{N}$. By [9, Cor. 5.5], x_0 is lucky for the denominator of $D_x^i(f)$ for all $i \in \mathbb{N}$. Then, the claim on $r_{0,i}$ follows from Lemma 10 applied to $D_x^i(f)$. Let ρ be the minimal order of a telescoper, then $r_0, \dots, r_\rho$ are linearly dependent over $k(x)$ by Lemma 13. Thus $r_{0,0}, \dots, r_{0,\rho}$ are linearly dependent over k , which implies $\rho_0 \leq \rho$. $\square$

3.1.2 Degree bounds for minimal telescopers

To derive degree bounds for g_i and r_i in (8), let δ, δ', μ , and μ' be defined as before Lemma 7, and set $\mu'' = \mu + \mu' - 1$.

Lemma 16 *Let W be in $k[x, y]$ with $\deg_y(W) < d_y^*$. Then, for all $i \in \mathbb{N}$, there exist $B, b \in k[x, y]$ with both $\text{bideg}(B)$ and $\text{bideg}(b)$ bounded by $(\deg_x(W) + \mu'', d_y^* - 1)$, such that*

$$D_x \left(\frac{W}{\delta^{i+1} \delta'^i Q^*} \right) = D_y \left(\frac{B}{\delta^{i+2} \delta'^{i+1} Q^*} \right) + \frac{b}{\delta^{i+2} \delta'^{i+1} Q^*}.$$

PROOF. A straightforward calculation leads to

$$D_x \left(\frac{W}{\delta^{i+1} \delta'^i Q^*} \right) = \frac{\tilde{W}}{\delta^{i+2} \delta'^{i+1} Q^*} - \frac{1}{\delta^{i+1} \delta'^i} \frac{W D_x(Q^*)}{Q^{*2}},$$

where $\text{bideg}(\tilde{W}) \leq (\deg_x(W) + \mu'', d_y^* - 1)$. By Corollary 8, there exist $\tilde{B}, \tilde{b} \in k[x, y]$ such that

$$\frac{1}{\delta^{i+1} \delta'^i} \frac{W D_x(Q^*)}{Q^{*2}} = \frac{1}{\delta^{i+2} \delta'^{i+1}} \left(D_y \left(\frac{\delta \tilde{B}}{Q^*} \right) + \frac{\delta \tilde{b}}{Q^*} \right),$$

with $\text{bideg}(\tilde{B})$ and $\text{bideg}(\tilde{b})$ bounded by $(\deg_x(W) + \mu' - 1, d_y^* - 1)$. Setting $(B, b) = (-\delta \tilde{B}, \tilde{W} - \delta \tilde{b})$ ends the proof. $\square$

Lemma 17 *For $i \in \mathbb{N}$, there exist $B_i, b_i \in k[x, y]$ such that*

$$D_x^i(f) = D_y \left(\frac{B_i}{\delta^{i+1} \delta'^i Q^{*i} Q^-} \right) + \frac{b_i}{\delta^{i+1} \delta'^i Q^*}. \quad (10)$$

Moreover, $\text{bideg}(B_i) \leq (\deg_x(P) + \mu + i\mu'' + (i-1)d_x^*, id_y^* + d_y^- - 1)$ and $\text{bideg}(b_i) \leq (\deg_x(P) + \mu + i\mu'' - d_x^*, d_y^* - 1)$.

PROOF. We proceed by induction on i . For $i = 0$, the claim follows from Lemma 7. Assume that $i > 0$ and that the claim holds for the values less than i . For brevity, we set $\gamma = \deg_x(P) + \mu$, $F_{i-1} = B_{i-1}/(\delta^i \delta'^{i-1} Q^{*i-1} Q^-)$, and $G_{i-1} = b_{i-1}/(\delta^i \delta'^{i-1} Q^*)$. The induction hypothesis implies

$$D_x^i(f) = D_y D_x(F_{i-1}) + D_x(G_{i-1}),$$

with bidegree bounds on B_{i-1} and b_{i-1} . Fact 3(i) implies that $\tilde{Q} := Q^* D_x(Q^-)/Q^-$ is in $k[x, y]$, with $\text{bideg}(\tilde{Q}) \leq$

Algorithm HermiteTelescoping(f)

INPUT: $f = P/Q \in k(x, y)$ satisfying Hypothesis (H).

OUTPUT: A minimal telescoper $L \in k[x]\langle D_x \rangle$ with certificate $g \in k(x, y)$.

1. Apply **HermiteEvalInterp** to f to get (g_0, a_0) such that $f = D_y(g_0) + a_0/Q^*$. If $a_0 = 0$, return $(1, g_0)$.
2. For i from 1 to $\deg_y(Q^*)$ do
 - (a) Apply **HermiteEvalInterp** to $-a_{i-1} D_x(Q^*)/Q^{*2}$ to express it as $D_y(\tilde{g}_i) + \tilde{a}_i/Q^*$.
 - (b) Set $g_i = D_x(g_{i-1}) + \tilde{g}_i$ and $a_i = D_x(a_{i-1}) + \tilde{a}_i$.
 - (c) Solve $\sum_{j=0}^i \eta_j a_j = 0$ for $\eta_j \in k(x)$ using [17]. If there exists a nontrivial solution, then set $(L, g) := (\sum_{j=0}^i \eta_j D_x^j, \sum_{j=0}^i \eta_j g_j)$, and break.
3. Compute the content c of L and return $(c^{-1}L, c^{-1}g)$.

Figure 3: Creative telescoping by Hermite reduction

$(d_x^* - 1, d_y^*)$. Hence $D_x(1/Q^-) = -\tilde{Q}/Q$. This observation and an easy calculation imply that

$$D_x(F_{i-1}) = \frac{\tilde{B}_{i-1}}{\delta^{i+1} \delta'^i Q^{*i} Q^-},$$

where $\tilde{B}_{i-1} \in k[x, y]$ and $\deg_x(\tilde{B}_{i-1}) \leq \deg_x(B_{i-1}) + \mu'' + d_x^*$. Furthermore, by Lemma 16 there are $\tilde{B}_i, \tilde{b}_i \in k[x, y]$ with bidegrees at most $(\deg_x(b_{i-1}) + \mu'', d_y^* - 1)$, such that

$$D_x(G_{i-1}) = D_y \left(\frac{\tilde{B}_i}{\delta^{i+1} \delta'^i Q^*} \right) + \frac{\tilde{b}_i}{\delta^{i+1} \delta'^i Q^*}.$$

Setting $B_i = \tilde{B}_{i-1} + \tilde{B}_i Q^{*i-1} Q^-$ and $b_i = \tilde{b}_i$, we arrive at (10). It remains to verify the degree bounds. The induction hypothesis implies that both $\deg_x(\tilde{B}_i)$ and $\deg_x(b_i)$ are bounded by $\gamma + i\mu'' - d_x^-$. It follows that $\deg_x(\tilde{B}_i Q^{*i-1} Q^-)$ is bounded by $\gamma + i\mu'' + (i-1)d_x^*$. Similarly, $\deg_x(\tilde{B}_{i-1})$ is bounded by $\gamma + i\mu'' + (i-1)d_x^*$, and so is $\deg_x(B_i)$. The bounds on degrees in y are obvious. $\square$

We next derive degree bounds for the minimal telescoper obtained at an intermediate stage of **HermiteTelescoping**; refined bounds on the output will be given by Theorem 25.

Lemma 18 *Under (H'), Step 2(c) of Algorithm HermiteTelescoping computes a minimal telescoper $L \in k[x]\langle D_x \rangle$ with order ρ and a certificate $g \in k(x, y)$ for P/Q with $\deg_x(L) \in \mathcal{O}(d_x d_y \rho^2)$ and $\text{bideg}(g) \in \mathcal{O}(d_x d_y \rho^2) \times \mathcal{O}(d_y \rho)$.*

PROOF. By Lemma 13, we exhibit a minimal telescoper by considering the first nontrivial linear dependence among the a_i 's in (10). Let M be the coefficient matrix of the system in (η_i) obtained from $\sum_{i=0}^\rho \eta_i a_i = 0$. By Lemma 17, M is of size at most $(\rho + 1) \times d_y^*$ and with coefficients of degree at most $\sigma := d_x + \mu + \rho\mu'' - d_x^-$ in x . Hence, there exists a solution $(\eta_0, \dots, \eta_\rho) \in k[x]^{\rho+1}$ of degree at most $\sigma\rho$ in x by Fact 2(ii). Since $\mu, \mu'' \in \mathcal{O}(d_x d_y)$ and $d_y^* \leq d_y$, the degree estimates of L and g are as announced. $\square$

3.1.3 Complexity estimates

We proceed to analyse the complexity of the algorithm in Figure 3 and of an optimisation.

Theorem 19 *Under Hyp. (H'), Algorithm HermiteTelescoping in Figure 3 is correct and takes $\tilde{O}(\rho^{\omega+1} d_x d_y^2)$ ops, where ρ is the order of the minimal telescoper.*

PROOF. The formulas in Step 2(a) create the loop invariant $D_x^i(f) = D_y(g_i) + a_i/Q^*$. Correctness then follows from Lemmas 12 and 20. Step 1 takes $\tilde{O}(d_x d_y^2)$ ops by Theorem 11 under (H'). By Lemma 17, $\deg_x(-a_{i-1} D_x(Q^*)) \in \mathcal{O}(id_x d_y)$. So the cost for performing Hermite reduction on $-a_{i-1} D_x(Q^*)/Q^{*2}$ in Step 2(a) is $\tilde{O}(id_x d_y^2)$ ops by Theorem 11. The bidegrees of g_i and a_i in Step 2(b) are in $\mathcal{O}(id_x d_y) \times \mathcal{O}(id_y)$ by Lemma 17. Since adding and differentiating have linear complexity, Step 2(b) takes $\tilde{O}(i^2 d_x d_y^2)$ ops. For each i , the coefficient matrix of $\sum_{j=0}^i \eta_j a_j = 0$ in Step 2(c) is of size at most $(i+1) \times d_y^*$ and with coefficients of degree at most $\deg_x(a_i) \in \mathcal{O}(id_x d_y)$. Moreover, the rank of this matrix is either i or $i+1$. Then, Step 2(c) takes $\tilde{O}(i^\omega d_x d_y^2)$ ops by Fact 2(iii). Computing the content and divisions in Step 3 has complexity $\tilde{O}(d_x d_y \rho^3)$. If the algorithm returns when $i = \rho$, then the total cost is in

$$\sum_{i=0}^{\rho} \tilde{O}(i^2 d_x d_y^2) + \sum_{i=1}^{\rho} \tilde{O}(i^\omega d_x d_y^2) \subset \tilde{O}(\rho^{\omega+1} d_x d_y^2) \text{ ops,} \quad (11)$$

which is as announced. $\square$

An optimisation, based on Lemma 15, consists in guessing the order ρ so as to perform Step 2(c) a few times only: As a preprocessing step, choose $x_0 \in k$ lucky for Q , then detect linear dependence of $\{r_{0,0}, \dots, r_{0,j}\}$ in (9). The minimal j for dependence is a lower bound ρ_0 on ρ . So Step 2(c) is then performed only when $i \geq \rho_0$. In practice, the lower bound ρ_0 computed in this way almost always coincides with the actual order ρ . So normalising the g_i 's becomes the dominant step, as observed in experiments. We analyse this optimisation by first estimating the cost for computing ρ_0 .

Lemma 20 *Under Hypothesis (H'), computing a lower order bound ρ_0 for minimal telescopers takes $\tilde{O}(d_x d_y \rho_0^3)$ ops.*

PROOF. Since differentiating has linear complexity, the derivative $D_x^i(f)$ takes $\tilde{O}(i^2 d_x d_y)$ ops. By Fact 1(i), the evaluation $D_x^i(f)(x_0, y)$ takes as much. The cost of Hermite reduction on $D_x^i(f)(x_0, y)$ is $\tilde{O}(id_y)$ ops by Lemma 5. By Fact 2(iii) with $d = 1$, computing the rank of the coefficient matrix of $\sum_{j=0}^i \eta_j r_{0,j}$, with $r_{0,j}$ as in (9), takes $\tilde{O}(d_y i^{\omega-1})$ ops. Thus, the total cost for computing a lower bound on ρ_0 is $\sum_{i=0}^{\rho_0} \tilde{O}(i^2 d_x d_y) \in \tilde{O}(d_x d_y \rho_0^3)$ ops. $\square$

Corollary 21 *For runs such that $\rho_0 = \rho - \mathcal{O}(1)$, the previous optimisation of HermiteTelescoping takes $\tilde{O}(\rho^3 d_x d_y^2)$ ops.*

PROOF. In view of Lemma 20, the estimate (11) becomes $\tilde{O}(d_x d_y \rho_0^3) + \sum_{i=0}^{\rho} \tilde{O}(i^2 d_x d_y^2) + \sum_{i=\rho_0}^{\rho} \tilde{O}(i^\omega d_x d_y^2)$, which is $\tilde{O}(\rho^3 d_x d_y^2) + \tilde{O}((\rho - \rho_0) \rho^\omega d_x d_y^2)$ ops, whence the result. $\square$

3.2 Almkvist and Zeilberger's approach

We analyse the complexity of Almkvist and Zeilberger's algorithm [1] when restricted to bivariate rational functions. In order to get a telescoper whose order ρ is minimal, the resulting algorithm, denoted RatAZ, solves (1) for increasing, prescribed values of ρ until it gets a solution $(\eta_0, \dots, \eta_\rho, g) \in k(x)^{\rho+1} \times k(x, y)$ with the η_i 's not all zero. For the analysis, we start by studying the parameterisation of the differential Gosper algorithm of [1] under the same restriction to $k(x, y)$.

Definition ([9]) *Let K be a field and $a, b \in K[y]$ be non-zero polynomials. A triple $(p, q, r) \in K[y]^3$ is said to be a differential Gosper form of the rational function a/b if*

$$\frac{a}{b} = \frac{D_y(p)}{p} + \frac{q}{r} \text{ and } \gcd(r, q - \tau D_y(r)) = 1 \text{ for all } \tau \in \mathbb{N}.$$

For hyperexponential f , a key step in [1] is to compute a differential Gosper form of the logarithmic derivative of $F = \sum_{i=0}^{\rho} \eta_i D_x^i(f)$, where the η_i 's are undetermined from $k(x)$. In the analogue RatAZ, this form is predicted by Lemma 22 below, which is a technical generalisation of a result by Le [12] on F when f has a squarefree denominator.

Write $Q = t(y)T(x, y)$, splitting content and primitive part w.r.t. x . By an easy induction, $D_x^i(f) = N_i/(QT^{*i})$ for $N_i \in k[x, y]$. For this section, set $F = \sum_{i=0}^{\rho} \eta_i D_x^i(f)$, $N = \sum_{i=0}^{\rho} \eta_i N_i T^{*\rho-i}$, and $H = -D_y(Q)/Q - \rho t^* D_y(T^*)$.

Lemma 22 *If F is nonzero, the triple (N, H, Q^*) is a differential Gosper form of $D_y(F)/F$.*

PROOF. First, observe $F = N/(QT^{*\rho})$ and $Q^* = t^* T^*$. Next, $D_y(F)/F = D_y(N)/N - D_y(Q)/Q - \rho D_y(T^*)/T^*$ is $D_y(N)/N + H/Q^*$. There remains to prove $\gcd(Q^*, H - \tau D_y(Q^*)) = 1$, for any $\tau \in \mathbb{N}$. Recall that the squarefree part Q^* of Q is the product $Q_1 Q_2 \dots Q_m$ and that $\hat{Q}_i$ denotes Q^*/Q_i . By Fact 3(ii),

$$Z := H - \tau D_y(Q^*) = -\rho t^* D_y(T^*) - \sum_{i=1}^m (i + \tau) \hat{Q}_i D_y(Q_i).$$

If Q_j divides t^* , Z reduces to $-(j + \tau) \hat{Q}_j D_y(Q_j)$ modulo Q_j . If not, it reduces to $-(j + \tau) \hat{Q}_j D_y(Q_j) - \rho t^* (D_y(Q_j) T^*/Q_j)$, which rewrites to $-(j + \tau + \rho) \hat{Q}_j D_y(Q_j)$ modulo Q_j . In both cases, Z is coprime with Q^* , as $j > 0$, $\tau \geq 0$, and $\rho \geq 0$. $\square$

By another induction, we observe $\text{bideg}(N_i) \leq (\deg_x(P) + i \deg_x(T^*) - i, d_y + i \deg_y(T^*) - 1)$, so that $\text{bideg}(N) \leq (\deg_x(P) + \rho \deg_x(T^*) - \rho, d_y + \rho \deg_y(T^*) - 1)$.

The next step in RatAZ is, for fixed ρ , to reduce (1) by the change of unknown $g = z/(Q^- T^{*\rho})$, so as to determine all $(\eta_i) \in k(x)^{\rho+1}$ for which the differential equation in z

$$\sum_{i=0}^{\rho} \eta_i N_i T^{*\rho-i} = Q^* D_y(z) + (D_y(Q^*) + H) z \quad (12)$$

has a polynomial solution in $k(x)[y]$. For later use, we recall the following consequence of [9, Corollary 9.6].

Lemma 23 *Let $a, b \in K[y]$ be such that $\beta = -\text{lc}_y(b)/\text{lc}_y(a)$ is a nonnegative integer and $\deg_y(b) = \deg_y(a) - 1$. Let $c \in K[y]$ be such that $\beta \geq \deg_y(c) - \deg_y(a) + 1$. If u is a polynomial solution of $a D_y(z) + bz = c$, then $\deg_y(u) \leq \beta$.*

The following lemma generalises [12, Lemma 2] to present a degree bound for z .

Lemma 24 *If $u \in k(x)[y]$ is a solution of (12) for $(\eta_i) \in k(x)^{\rho+1}$, then $\deg_y(u)$ is bounded by $\beta = d_y^- + \rho \deg_y(T^*)$.*

PROOF. Let $a = Q^*$ and $b = D_y(Q^*) + H$. By the definition of H , $b = -Q^* D_y(Q^-)/Q^- - \rho t^* D_y(T^*)$. Fact 3(i) implies that $\text{lc}_y(b) = -(d_y^- + \rho \deg_y(T^*)) \text{lc}_y(a)$. Therefore, $\beta = -\text{lc}_y(b)/\text{lc}_y(a) = d_y^- + \rho \deg_y(T^*)$. As $\deg_y(N) < d_y + \rho \deg_y(T^*)$ and $d_y = d_y^* + d_y^-$, $\beta \geq \deg_y(N) - d_y^* + 1$. The lemma holds by Lemma 23. $\square$

Algorithm RatAZ(f)

INPUT: $f = P/Q \in k(x, y)$ satisfying Hypothesis (H).

OUTPUT: A minimal telescoper $L \in k[x]\langle D_x \rangle$ with certificate $g \in k(x, y)$.

1. Compute $Q^- = \gcd(Q, D_y(Q))$, $Q^* = Q/Q^-$, and T, T^* primitive parts of Q, Q^* w.r.t. x , respectively;
2. Set $(\tilde{N}, N, \beta, H)$ to $(P, P, d_y^-, -Q^* D_y(Q)/Q)$;
3. For $\ell = 0, 1, \dots$ do
 - (a) Set z to $\sum_{j=0}^{\beta} z_j y^j$, extract the linear system $\mathcal{M}(\eta_i z_j)^T = 0$ from (12) (for $\rho = \ell$) and compute a basis S of the null space of $\mathcal{M}$ by [17].
 - (b) If S contains a solution $(\eta_0, \dots, \eta_\ell, s)$ such that $\eta_0, \dots, \eta_\ell$ are not all nonzero, then set $(L, g) := (\sum_{i=0}^{\ell} \eta_i D_x^i, s/(Q^- T^{*\ell}))$, and go to Step 4;
 - (c) Update $\tilde{N} := D_x(\tilde{N})T^* - \tilde{N}(T^* D_x(T)/T + i D_x(T^*))$, $N := NT^* + \eta_{\ell+1} \tilde{N}$, $\beta := \beta + \deg_y(T^*)$, and $H := H - t^* D_y(T^*)$.
4. Compute the content c of L and return $(c^{-1}L, c^{-1}g)$.

Figure 4: Improved Almkvist–Zeilberger algorithm

We end the present section using the approach of Almkvist and Zeilberger to provide tight degree bounds on the outputs from Algorithms HermiteTelescoping and RatAZ.

Theorem 25 *Under Hypothesis (H'), there exists a minimal telescoper $L \in k[x]\langle D_x \rangle$ with certificate $g \in k(x, y)$ with $\deg_x(L) \in \mathcal{O}(d_x d_y d_y^*)$ and $\text{bideg}(g) \in \mathcal{O}(d_x d_y d_y^*) \times \mathcal{O}(d_y d_y^*)$.*

PROOF. By Corollary 14, there exists a smallest $\rho \in \mathbb{N}$ at most d_y^* , for which (1) has a solution with the η_i 's not all zero. For this ρ , we estimate the size of the polynomial matrix $\mathcal{M}$ derived from (12) by undetermined coefficients. By the remark on N after Lemma 22, we have $\text{bideg}(N) \leq (n_x, n_y)$ where $n_x := d_x + \rho \deg_x(T^*) - \rho \in \mathcal{O}(\rho d_x)$ and $n_y := d_y + \rho \deg_y(T^*) - 1 \in \mathcal{O}(\rho d_y)$. The matrix $\mathcal{M}$ contains two blocks $\mathcal{M}_1 \in k[x]_{\leq n_x}^{(n_y+1) \times (\rho+1)}$ and $\mathcal{M}_2 \in k[x]_{\leq d_x}^{(n_y+1) \times (\beta+1)}$, where $\beta \in \mathcal{O}(\rho d_y)$ is the same as in Lemma 24. By the minimality of ρ , the dimension of the null space of $\mathcal{M}$ is 1. So there exists $u \in k[x]^{n_y+1}$ with coefficients of degree at most $n_x(\rho+1) + d_x(\beta+1) \in \mathcal{O}(d_x d_y d_y^*)$ in x such that $\mathcal{M}(\eta z)^T = 0$, which implies degree bounds in x for L and g . The degree bound in y for g is obvious. $\square$

We now analyse the complexity of the algorithm in Fig. 4.

Theorem 26 *Under Hypothesis (H'), Algorithm RatAZ in Figure 4 is correct and takes $\tilde{\mathcal{O}}(d_x d_y^\omega \rho^{\omega+2})$ ops, where ρ is the order of the minimal telescoper.*

PROOF. By the existence of a telescoper, Corollary 14, and Lemma 24, the algorithm always terminates and returns a minimal telescoper L , of order ρ at most d_y^* . Gcd computations dominate the cost of Steps 1 and 2, which take $\tilde{\mathcal{O}}(d_x d_y^2)$ ops. For each $\ell \in \mathbb{N}$, the dominating cost in Step 3 is computing the null space of $\mathcal{M}$. Let $n_y = d_y + \ell \deg_y(T^*) - 1 \in \mathcal{O}(\ell d_y)$ and $n_x = d_x + \ell \deg_x(T^*) \in \mathcal{O}(\ell d_x)$. By the same argument as in the proof of Theorem 25, the matrix $\mathcal{M}$ is of size at most $(n_y + 1) \times (\ell + \beta + 2)$ and with coefficients of degree at

most n_x . Let r be the rank of $\mathcal{M}$, which is either $\ell + \beta + 2$ or $\ell + \beta + 1$ by construction. Thus, a basis of the null space of $\mathcal{M}$ can be computed within $\tilde{\mathcal{O}}(n_x(n_y + 1)(\ell + \beta + 2)r^{\omega-2})$ ops by Fact 2(iii). Since $\beta \in \mathcal{O}(\ell d_y)$, $\tilde{\mathcal{O}}(n_x(n_y + 1)(\ell + \beta + 2)r^{\omega-2})$ is included in $\tilde{\mathcal{O}}(d_x d_y^\omega \ell^{\omega+1})$. Since Step 3 terminates at $\ell = \rho$, the total cost of the algorithm is $\sum_{\ell=0}^{\rho} d_x d_y^\omega \ell^{\omega+1}$ ops. This is within the announced complexity, $\tilde{\mathcal{O}}(d_x d_y^\omega \rho^{\omega+2})$ ops. $\square$

Corollary 27 *Algorithms HermiteTelescoping and RatAZ in Fig. 3 and 4 both output the primitive minimal telescoper L together with its certificate g , which satisfy $\deg_{D_x}(L) \leq d_y^*$, $\deg_x(L), \deg_x(g) \in \mathcal{O}(d_x d_y d_y^*)$, and $\deg_y(g) \in \mathcal{O}(d_y d_y^*)$.*

PROOF. Both algorithms output the primitive minimal telescoper, as they compute a minimal telescoper at an intermediate step, and owing to their last step of content removal. Bounds follow from Corollary 14 and Theorem 25. $\square$

4. NONMINIMAL TELESCOPERS

Here, we discard Hypothesis (H) and trade the minimality of telescopers for smaller total output sizes. To this end, we adapt and slightly extend the arguments in [13] and [3, §3].

Given $f = P/Q \in k(x, y)$ of bidegree (d_x, d_y) , our goal is to find a (possibly nonminimal) telescoper for f . It is sufficient to find a nonzero differential operator $A(x, D_x, D_y)$ that annihilates f . Indeed, any $A \in k[x]\langle D_x, D_y \rangle \setminus \{0\}$ such that $A(f) = 0$ can be written $A = D_y^r(L + D_y R)$, where L is nonzero in $k[x]\langle D_x \rangle$ and $R \in k[x]\langle D_x, D_y \rangle$. If $r = 0$, then clearly L is a telescoper for f ; otherwise, $A(f) = 0$ yields $L(f) = D_y(-R(f) - \sum_{i=0}^{r-1} \frac{a_i}{i+1} y^{i+1})$ for some $a_i \in k(x)$, which implies that L is again a telescoper for f . Moreover, in both cases, $\deg_x(L) \leq \deg_x(A)$ and $\deg_{D_x}(L) \leq \deg_{D_x}(A)$. Furthermore, for any $(i, j, \ell) \in \mathbb{N}^3$, a direct calculation yields

$$x^i D_x^j D_y^\ell(f) = \frac{H_{i,j,\ell}}{Q^{j+\ell+1}}, \quad (13)$$

where $H_{i,j,\ell} \in k[x, y]$ and $\deg_x(H_{i,j,\ell}) \leq (j + \ell + 1)d_x + i - j$ and $\deg_y(H_{i,j,\ell}) \leq (j + \ell + 1)d_y - \ell$. From these inequalities, we derive the size and complexity estimates in Figure 1 (bottom half), using two different filtrations of $k[x]\langle D_x, D_y \rangle$.

Lipshitz's filtration ([13]). Let F_ν be the k -vector space of dimension $\mathfrak{f}_\nu := \binom{\nu+3}{3}$ spanned by $\{x^i D_x^j D_y^\ell \mid i + j + \ell \leq \nu\}$. By (13), $F_\nu(f)$ is contained in the vector space of dimension $\mathfrak{g}_\nu := ((\nu + 1)d_x + \nu + 1)((\nu + 1)d_y + 1)$ spanned by $\{\frac{x^i y^j}{Q^{\nu+1}} \mid i \leq (\nu + 1)d_x + \nu, j \leq (\nu + 1)d_y\}$. Choosing $\nu = 6(d_x + 1)(d_y + 1)$ yields $\mathfrak{f}_\nu > \mathfrak{g}_\nu$; therefore, there exists A in $k\langle x, D_x, D_y \rangle \setminus \{0\}$ with total degree at most $6(d_x + 1)(d_y + 1)$ in x, D_x , and D_y that annihilates f . Moreover, A is found by linear algebra in dimension $\mathcal{O}((d_x d_y)^3)$.

A better filtration ([3]). Instead of taking total degree, set $F_{\kappa,\nu}$ to the k -vector space of dimension $\mathfrak{f}_{\kappa,\nu} := (\kappa + 1)\binom{\nu+2}{2}$ generated by $\{x^i D_x^j D_y^\ell \mid i \leq \kappa, j + \ell \leq \nu\}$. By (13), $F_{\kappa,\nu}(f)$ is contained in the vector space of dimension $\mathfrak{g}_{\kappa,\nu} := ((\nu + 1)d_x + \kappa + 1)((\nu + 1)d_y + 1)$ spanned by $\{\frac{x^i y^j}{Q^{\nu+1}} \mid i \leq (\nu + 1)d_x + \kappa, j \leq (\nu + 1)d_y\}$. Choosing $\kappa = 3d_x d_y$ and $\nu = 6d_y$ results in $\mathfrak{f}_{\kappa,\nu} > \mathfrak{g}_{\kappa,\nu}$. This implies the existence of A in $k\langle x, D_x, D_y \rangle \setminus \{0\}$ with total degree at most $6d_y$ in D_x and D_y and degree at most $3d_x d_y$ in x that annihilates f . Again, A is found by linear algebra over k , but in smaller dimension $\mathcal{O}(d_x d_y^3)$.

No.	AZ	Abr	RAZ	H1	H2	HO	EI	MG
29	44	72	32	28	36	20	608	528
43	52	76	36	20	24	32	652	584
46	4268	1436	784	492	1288	752	343413	18945
49	474269	34694	20977	10336	36254	22417	∞	652968

Table 1: Creative telescoping on random instances
Timings in ms for algorithms in Table 3 (stopped after 30 min).

5. IMPLEMENTATION AND TIMINGS

We implemented in Maple 13 all the algorithms described; as we used Maple’s generic solver `SolveTools:-Linear`, all of our implementations are deterministic.

The evaluation-interpolation algorithm `HermiteEvalInterp` for Hermite reduction (Fig. 2) does not perform well, mainly because Maple’s rational interpolation routines are far too slow. We thus implemented Algorithm `HermiteReduce` (original version) in [4, § 2.2] (carefully avoiding redundant extended gcd calculations), and noted that it performs better.

We then implemented a variant of Algorithm `HermiteTelescoping` in Figure 3, using `HermiteReduce` in place of `HermiteEvalInterp`, and including the optimisation at the end of § 3.1.3, refined by additional modular calculations.

For a rational function, Algorithm `HermiteTelescoping` returns the minimal telescoper L and the certificate g . The algorithm separates the computation for L from that for g . Indeed, g is formed by the coefficients of L , g_0 , the $\tilde{g}_i$ and their derivatives given in Figure 3. This feature enables us to either return the certificate g as a sum of unnormalised rational functions, or a normalised rational function.

A selection of timings by this implementation and others are given in Table 1; our code, the full table, as well as the random inputs are given in [20]. For our experiments, we exhaustively considered all 49 bidegree patterns in factorisations of denominators $Q_1 \cdots Q_m^m$ ($m \leq 5$) that add up to bidegree (5,5), and generated corresponding random denominators, imposing the integers of the expanded forms to have around 26 digits. Numerators were generated as random bidegree-(5,5) polynomials with coefficients of 26 digits.

Application to diagonals. The diagonal of a formal power series $f = \sum_{i,j \geq 0} f_{i,j} x^i y^j$ in $k[[x, y]]$ is defined to be the power series $\Delta(f) := \sum_{i=0}^{\infty} f_{i,i} x^i$. For a D-finite power series f , it is known to be D-finite [13], and it is even algebraic for a bivariate rational function $f \in k(x, y) \cap k[[x, y]]$ [16, § 6.3]. A linear differential operator $L \in k(x) \langle D_x \rangle$ that annihilates $\Delta(f)$ can then be computed via rational-function telescoping, owing to the following classical lemma from [13].

Lemma 28 *Any telescoper for $f(y, \frac{x}{y})/y$ annihilates $\Delta(f)$.*

By this lemma, it suffices to compute a telescoper without its certificate to get an annihilator. Algorithm `HermiteTelescoping` is suitable for this task, since it separates computation of telescopers and certificates. Alternatively, for $f = P/Q$, we can compute an annihilator of $\Delta(f)$ either as the differential resolvent of the resultant $\text{Res}_y(Q, P - \tau D_y Q)$, or simply guess it from the first terms of the series expansion of $\Delta(f)$.

We compare the various algorithms on an example borrowed from [5] (timings of execution are given in Table 2):

$$f = \frac{1}{1 - x - y - xy(1 - x^d)}, \text{ where } d \in \mathbb{N}. \quad (14)$$

All computer calculations have been performed on a Quad-Core Intel Xeon X5482 processor at 3.20GHz, with 3GB of RAM, using up to 6.5GB of memory allocated by Maple.

d	AZ	Abr	RAZ	H1	H2	HO	RR	GHP
4	176	136	100	116	208	108	220	956
8	3032	4244	4380	1976	5344	4396	10336	154409
10	11740	12816	7108	7448	24565	7076	46882	1118313
4	184	168	120	120	220	116	224	1340
8	3540	3704	2540	2092	6976	2516	10348	271480
10	16817	17013	9200	8068	32218	9092	46750	∞

Table 2: Computation of the diagonals of (14)

Timings in ms by creative telescoping of $f(y, x/y)/y$ (upper half) or $f(y/x, x)/x$ (second half). Algorithms listed in Table 3.

AZ `DETools[Zeilberger]`
Abr AZ with Abramov’s denominator bound by option `gasper_free`
RAZ Algorithm `RatAZ` of Fig. 4, with lower-bound prediction
H1 our Hermite-based approach, without certificate normalisation
H2 H1, but with normalised certificate
HO RAZ, solving (1) by Horowitz–Ostrogradsky
EI H1 with evaluation and interpolation for calculations over $k(x)$
MG `Mgfun`’s creative telescoping for general D-finite functions
RR telescoper computation by resultant and differential resolvent
GHP telescoper guessing by diagonal expansion and Hermite–Padé

Table 3: List of the algorithms for the experiments

6. REFERENCES

- [1] G. Almkvist and D. Zeilberger. The method of differentiating under the integral sign. *J. Symb. Comput.*, 10:571–591, 1990.
- [2] M. Apagodu and D. Zeilberger. Multi-variable Zeilberger and Almkvist-Zeilberger algorithms and the sharpening of Wilf-Zeilberger theory. *Adv. in Appl. Math.*, 37(2):139–152, 2006.
- [3] A. Bostan, F. Chyzak, B. Salvy, G. Lecerf, and É. Schost. Differential equations for algebraic functions. In *ISSAC’07*, pages 25–32. ACM, New York, 2007.
- [4] M. Bronstein. *Symbolic Integration I: Transcendental functions*, volume 1 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, second edition, 2005.
- [5] A. Flaxman, A. W. Harrow, and G. B. Sorkin. Strings with maximally many distinct subsequences and substrings. *Electron. J. Combin.*, 11(1):R8, 10 pp., 2004.
- [6] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, second edition, 2003.
- [7] K. O. Geddes and H. Q. Le. An algorithm to compute the minimal telescopers for rational functions (differential-integral case). In *Mathematical Software*, pages 453–463. WSP, 2002.
- [8] J. Gerhard. Fast modular algorithms for squarefree factorization and Hermite integration. *Appl. Algebra Engrg. Comm. Comput.*, 11(3):203–226, 2001.
- [9] J. Gerhard. *Modular Algorithms in Symbolic Summation and Symbolic Integration (LNCS)*. SpringerVerlag, 2004.
- [10] C. Hermite. Sur l’intégration des fractions rationnelles. *Ann. Sci. École Norm. Sup. (2)*, 1:215–218, 1872.
- [11] E. Horowitz. Algorithms for partial fraction decomposition and rational function integration. In *SYMSAC’71*, pages 441–457, New York, USA, 1971. ACM.
- [12] H. Q. Le. On the differential-integral analogue of Zeilberger’s algorithm to rational functions. In *Proc. of the 2000 Asian Symposium on Computer Mathematics*, pages 204–213, 2000.
- [13] L. Lipshitz. The diagonal of a D-finite power series is D-finite. *J. Algebra*, 113(2):373–378, 1988.
- [14] L. Lipshitz. D-finite power series. *J. Algebra*, 122(2):353–373, 1989.
- [15] M. Ostrogradsky. De l’intégration des fractions rationnelles. *Bull. de la classe physico-mathématique de l’Acad. Impériale des Sciences de Saint-Petersbourg*, 4:145–167, 286–300, 1845.
- [16] R. P. Stanley. *Enumerative Combinatorics. Vol. 2*, volume 62 of *Cambridge Studies in Advanced Mathematics*. CUP, 1999.
- [17] A. Storjohann and G. Villard. Computing the rank and a small nullspace basis of a polynomial matrix. In *ISSAC’05*, pages 309–316. ACM, New York, 2005.
- [18] D. Zeilberger. A holonomic systems approach to special functions identities. *J. Comput. Appl. Math.*, 32:321–368, 1990.
- [19] D. Zeilberger. The method of creative telescoping. *J. Symbolic Comput.*, 11(3):195–204, 1991.
- [20] <http://algo.inria.fr/chen/BivRatCT/>, 2010.

Partial Denominator Bounds for Partial Linear Difference Equations

Manuel Kauers^{*}
RISC
Johannes Kepler University
4040 Linz (Austria)
mkauers@risc.jku.at

Carsten Schneider[†]
RISC
Johannes Kepler University
4040 Linz (Austria)
cschneid@risc.jku.at

ABSTRACT

We investigate which polynomials can possibly occur as factors in the denominators of rational solutions of a given partial linear difference equation (PLDE). Two kinds of polynomials are to be distinguished, we call them *periodic* and *aperiodic*. The main result is a generalization of a well-known denominator bounding technique for univariate equations to PLDEs. This generalization is able to find all the aperiodic factors of the denominators for a given PLDE.

Categories and Subject Descriptors

I.1.2 [Computing Methodologies]: Symbolic and Algebraic Manipulation—*Algorithms*

General Terms

Algorithms

Keywords

Difference Equations, Rational Solutions

1. INTRODUCTION

Several algorithms in symbolic computation depend on a subroutine for finding the rational solutions of an ordinary linear difference (or differential) equation (OLDE), and several algorithms are known for implementing such a subroutine [1, 2, 4, 11, 13, 14, 6, 8, 9]. On a conceptual level, the typical approach for finding rational solutions can be divided into three steps. In the first step, one constructs a polynomial Q such that the denominator q of any potential solution p/q must divide Q . This polynomial Q is called *universal denominator* or *denominator bound*. In the second step, the

universal denominator is used to transform the given equation into a new equation such that P is a *polynomial* solution of the new equation if and only if P/Q is a *rational* solution of the original one. In the third and final step, the polynomial solutions P of the transformed equation are determined.

The first algorithm for computing universal denominators in the case of OLDEs with polynomial coefficients was proposed in 1971 by Abramov [1] (see Section 2 below for a summary). It has been generalized to q -difference equations [3], to matrix equations [5], and also to equations whose coefficients belong to domains other than polynomials. For example, Bronstein [7] and Schneider [12] have observed that a universal denominator can be constructed also when the coefficient domains are difference fields which can be used for representing nested sums and products ($\Pi\Sigma$ -fields). For such domains, the situation is more involved. There is a need to distinguish between “normal” factors of the universal denominator which can be found very much like in the usual polynomial case, and “special” factors which have to be constructed by some other means.

In the present article, we consider partial (i.e., multivariate) linear difference equations with polynomial coefficients (PLDEs). Our ultimate goal is the construction of a universal denominator for potential rational solutions of a given PLDE. Like in the univariate case with sophisticated coefficient domains, there are two kinds of factors to be distinguished. As a matter of fact, some parts of the denominator cannot be bound at all. For example, the equation

$$f(n+1, k) = f(n, k+1)$$

has $(n+k)^{-\alpha}$ as a rational solution, for any $\alpha \in \mathbb{N}$, and there is obviously no finite polynomial Q that would be a multiple of $(n+k)^\alpha$ for all $\alpha \in \mathbb{N}$. We will call factors that may exhibit such “special” behaviour *periodic*. Our main result is that we can construct for any given PLDE a polynomial d such that every *aperiodic* factor of any potential solution p/q must divide d .

Such a bound on the aperiodic factors of the denominators does not directly give rise to a full algorithm for finding rational solutions of PLDEs, but it can be considered as a step in this direction. For a full algorithm, besides of the bounding of the periodic parts of the denominator, also the entire question of how to find (all) polynomial solutions of a PLDE in the third step is wide open and far from being settled. But even if these parts have to remain open for now, our aperiodic denominator bound is useful in practice. When it comes to solving an actual equation, possible periodic factors in a

^{*}Supported by the Austrian Science Fund (FWF) grants P20347-N18 and Y464-N18.

[†]Supported by the Austrian Science Fund (FWF) grants P20162-N18 and P20347-N18.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

solution can often be guessed by inspection, their multiplicities can be determined by trial and error, and degree bounds for polynomial solutions can be established heuristically. A reasonably tight universal denominator, on the other hand, cannot be as easily obtained on heuristic grounds.

2. THE UNIVARIATE CASE

Before entering the multivariate setting, let us summarize Abramov's classical denominator bound for univariate equations. We will introduce on the fly some notions and notations needed later.

Let $\mathbb{K}$ be a field of characteristic zero and let $\mathbb{K}[n]$ and $\mathbb{K}(n)$ denote the ring of univariate polynomials and the field of rational functions in n with coefficients in $\mathbb{K}$, respectively. Write N for the shift operator acting on $\mathbb{K}[n]$ and $\mathbb{K}(n)$ via

$$Nq(n) := q(n+1).$$

The objects of interest are difference equations of the form

$$a_0y + a_1Ny + \dots + a_mN^my = f \quad (1)$$

where $a_0, \dots, a_m, f \in \mathbb{K}[n]$ ($a_0, a_m \neq 0$) are given and $y \in \mathbb{K}(n)$ is unknown.

The denominator bounding problem is as follows: given $a_0, \dots, a_m, f \in \mathbb{K}[n]$, find $Q \in \mathbb{K}[n] \setminus \{0\}$ such that the denominator of any solution $y \in \mathbb{K}(n)$ of (1) divides Q . Abramov's denominator bounding algorithm [1] is an efficient way of computing

$$\gcd\left(\prod_{i=0}^s N^i a_0, \prod_{i=0}^s N^{m-i} a_m\right), \quad (2)$$

where

$$s := \max\{i \geq 0 : \gcd(N^i a_0, N^{-m} a_m) \neq 1\} \quad (3)$$

is the *dispersion* of a_0 and $N^{-m} a_m$. It is efficient in the sense that the gcd is constructed without explicitly calculating the products.

To see that this bound is correct, write (1) in the form

$$N^m y = \frac{1}{a_m} \left(f - \sum_{i < m} a_i N^i y \right).$$

Shifting this equation by s gives

$$N^{m+s} y = \frac{1}{N^s a_m} \left(N^s f - \sum_{i < m} (N^s a_i) (N^{i+s} y) \right).$$

By repeatedly using the recurrence, the terms $N^{i+s} y$ appearing on the right hand side can be reduced to smaller shifts of y so that for certain polynomials $b, b_0, \dots, b_{m-1}$ we have

$$N^{m+s} y = \frac{b - \sum_{i < m} b_i N^i y}{\prod_{i=0}^s N^i a_m}.$$

At this point we rely on the following result.

THEOREM 1. [1] *For any solution $y = \frac{p}{q} \in \mathbb{K}(n)$ of (1),*

$$\max\{i \geq 0 : \gcd(q, N^i q) \neq 1\} \leq s.$$

This theorem ensures that the denominator of a solution y cannot contain two factors u, v with $u = N^{s+1}v$, and this in turn implies that no denominator of any of the $N^i y$ on the right can have a common factor with the denominator of $N^{m+s} y$. Therefore the denominator of $N^{m+s} y$ must be

a divisor of $\prod_{i=0}^s N^i a_m$, and therefore the denominator of y must be a divisor of

$$\prod_{i=0}^s N^{i-m-s} a_m = \prod_{i=0}^s N^{-m-i} a_m. \quad (4)$$

By an analogous argument, now rewriting y in terms of higher shifts, it can be shown that the denominator of y must divide $\prod_{i=0}^s N^i a_0$. As both bounds must hold simultaneously, the correctness of Abramov's bound is established. The argument given here is not exactly Abramov's original one, but follows a proof which to our knowledge first appeared in [5]. The equivalence of the two approaches is shown in [8].

3. THE MULTIVARIATE CASE

For the rest of this article we will be concerned with adapting the univariate reasoning sketched in the previous section to the multivariate setting. From now on, we consider polynomials and rational functions in r variables $n_1, \dots, n_r$. Wherever it seems appropriate, we will use multiindex notation, writing for instance $\mathbf{n}$ for $n_1, \dots, n_r$, etc.

We define shift operators $N_1, \dots, N_r$ acting on $\mathbb{K}[\mathbf{n}]$ and $\mathbb{K}(\mathbf{n})$ in the obvious way:

$$\begin{aligned} N_i q(n_1, \dots, n_r) \\ = q(n_1, \dots, n_{i-1}, n_i + 1, n_{i+1}, \dots, n_r). \end{aligned}$$

For $\mathbf{i} = (i_1, \dots, i_r) \in \mathbb{Z}^r$ we will abbreviate

$$N^{\mathbf{i}} q := N_1^{i_1} N_2^{i_2} \dots N_r^{i_r} q.$$

A partial linear difference equation (PLDE) is an equation of the form

$$\sum_{\mathbf{s} \in S} a_{\mathbf{s}} N^{\mathbf{s}} y = f, \quad (5)$$

where $S \subseteq \mathbb{Z}^r$ is finite and nonempty (called the *support* or the *shift set* or the *structure set* of the equation), $f \in \mathbb{K}[\mathbf{n}]$ and $a_{\mathbf{s}} \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ ($\mathbf{s} \in S$) are explicitly given polynomials, and $y \in \mathbb{K}(\mathbf{n})$ is an unknown rational function. The polynomial $a_{\mathbf{s}}$ is called the coefficient of $N^{\mathbf{s}}$ (or simply of $\mathbf{s}$).

In the following section, we generalize the notion of dispersion to multivariate polynomials, and we will define the notions of periodic and aperiodic polynomials. After that, in Section 5, we will show how to predict all the aperiodic factors of a rational solution y of (5).

4. SPREAD AND DISPERSION

The notions of spread and dispersion are related to the question whether two polynomials can be mapped to one another by a shift. In the multivariate case, we now allow independent shifts in all directions.

Given two polynomials $p, q \in \mathbb{K}[\mathbf{n}]$, we say that they are shift equivalent if there exists $\mathbf{i} = (i_1, \dots, i_r) \in \mathbb{Z}^r$ such that $p(\mathbf{n}) = q(\mathbf{n} + \mathbf{i})$. In operator notation, p and q are shift equivalent iff $p = N^{\mathbf{i}} q$.

DEFINITION 1. *Let $p, q \in \mathbb{K}[\mathbf{n}]$. The set*

$$\begin{aligned} \text{Spread}(p, q) := \{ (i_1, \dots, i_r) \in \mathbb{Z}^r : \\ \gcd(p, N_1^{i_1} \dots N_r^{i_r} q) \neq 1 \} \end{aligned}$$

is called the spread of p and q . The number

$$\text{Disp}_k(p, q) := \max\{ |i_k| : (i_1, \dots, i_r) \in \text{Spread}(p, q) \}$$

is called the dispersion of p and q w.r.t. $k \in \{1, \dots, r\}$, and

$$\text{Disp}(p, q) := \max(\text{Disp}_1(p, q), \dots, \text{Disp}_r(p, q))$$

is called dispersion of p and q . (By convention, $\max A := -\infty$ if A is empty and $\max A := \infty$ if A is unbounded.)

The polynomial p is called periodic if $\text{Spread}(p, p)$ is infinite and aperiodic otherwise.

Note that this definition does not exactly correspond to the definition stated before for the univariate case. While there, the definition depends on whether one shifts to the left or to the right [1, 7], our definition takes all directions into account. This makes the reasoning below a little simpler.

EXAMPLE 1. Let

$$\begin{aligned} p &= (n^2 + k^2)((n+1)^2 + (k-3)^2)(k-n+3), \\ q &= ((n+2)^2 + (k-1)^2)((n-2)^2 + (k+7)^2)(2k-3n). \end{aligned}$$

Then, by inspection,

$$\begin{aligned} \text{Spread}(p, q) &= \{(2, -1), (-2, 7), (1, 2), (-3, 10)\}, \\ \text{Disp}(p, q) &= 10. \end{aligned}$$

Both p and q are aperiodic. An example for a periodic polynomial is $n - k$, because, again by inspection,

$$\begin{aligned} \text{Spread}(n - k, n - k) &= \{(i, i) : i \in \mathbb{Z}\} \\ &= \{\dots, (-1, -1), (0, 0), (1, 1), (2, 2), \dots\} \end{aligned}$$

is infinite.

In the univariate case, the spread of two polynomials can be found as the set of all integer roots of the polynomial $\text{res}_n(p(n), q(n+i)) \in \mathbb{K}[i]$. This is no longer possible in the multivariate setting, for in the case of several variables, common roots no longer correspond to common factors. Nevertheless, it turns out that the multivariate spread as defined as above can be effectively computed. Let us consider the somewhat simpler situation of irreducible polynomials first. In this situation the spread cannot take on any cardinality:

LEMMA 1. Let $p, q \in \mathbb{K}[\mathbf{n}]$ be irreducible and aperiodic. Then $|\text{Spread}(p, q)| \leq 1$.

PROOF. Suppose p and q are such that $|\text{Spread}(p, q)| > 1$. Then there exist two different multiindices $(i_1, \dots, i_r)$ and $(j_1, \dots, j_r)$ with

$$\gcd(p, N_1^{i_1} \dots N_r^{i_r} q) \neq 1 \quad \text{and} \quad \gcd(p, N_1^{j_1} \dots N_r^{j_r} q) \neq 1.$$

As p and q are irreducible and irreducibility is preserved under the shifts $n_i \mapsto n_i + 1$, we have in fact

$$cp = N_1^{i_1} \dots N_r^{i_r} q = N_1^{j_1} \dots N_r^{j_r} q$$

for some $c \in \mathbb{K} \setminus \{0\}$, hence

$$q = N_1^{i_1-j_1} \dots N_r^{i_r-j_r} q,$$

in contradiction to the assumption that q is aperiodic. $\square$

It is not essential for the lemma that we consider only shifts of integer distance. More generally, if p and q are two irreducible polynomials, then, by the same argument, the number of vectors $\mathbf{i} \in \bar{\mathbb{K}}^r$ with $p(\mathbf{n}) = q(\mathbf{n} + \mathbf{i})$ is either 0 or 1 or infinite ($\bar{\mathbb{K}}$ refers to the algebraic closure of $\mathbb{K}$).

These vectors $\mathbf{i} \in \bar{\mathbb{K}}$ can be found by making a brute force ansatz. For a variable vector $\mathbf{i} = (i_1, \dots, i_r)$, force

$$p(\mathbf{n}) - q(\mathbf{n} + \mathbf{i}) \stackrel{!}{=} 0$$

and compare coefficients with respect to $\mathbf{n}$ to obtain an algebraic system of equations in $i_1, \dots, i_r$ over $\mathbb{K}$. The solutions will form an affine linear space over $\mathbb{K}$, for whenever $\mathbf{i}, \mathbf{j} \in \bar{\mathbb{K}}^r$ are such that $p(\mathbf{n}) = q(\mathbf{n} + \mathbf{i})$ and $p(\mathbf{n}) = q(\mathbf{n} + \mathbf{j})$, then

$$p(\mathbf{n}) = q(\mathbf{n} + \mathbf{i} + \alpha(\mathbf{i} - \mathbf{j})) \text{ for all } \alpha \in \mathbb{Z},$$

and since the solution set is Zariski-closed, what is true for all $\alpha \in \mathbb{Z}$ must also be true for all $\alpha \in \mathbb{K}$.

The spread of two irreducible polynomials can therefore be computed by first determining a basis of the affine linear space of all possible shifts $\mathbf{i} \in \bar{\mathbb{K}}^r$ mapping one given polynomial to the other. By taking the radical to remove nontrivial multiplicities, it is ensured that a basis of the linear space can be read off from a Gröbner basis. In a second step, we filter out from this affine space the vectors which have integral coordinates only:

ALGORITHM 1. Input: $p, q \in \mathbb{K}[\mathbf{n}]$ irreducible
Output: $\text{Spread}(p, q)$

- 1 $S := \text{Coefficients}(p(\mathbf{n}) - q(\mathbf{n} + \mathbf{i}), \{\mathbf{n}\}) \subseteq \mathbb{K}[\mathbf{i}]$
- 2 $G := \text{GröbnerBasis}(\text{Radical}(S), \text{degrevlex}(\mathbf{i}))$
- 3 Choose a basis B of the $\mathbb{Q}$ vector space generated by the coefficients of the elements of G , say $B = \{b_1, \dots, b_d\} \subseteq \mathbb{K}$.
- 4 For each $g \in G$, let $g^{(1)}, \dots, g^{(d)} \in \mathbb{Q}[\mathbf{i}]$ be such that $g = b_1 g^{(1)} + b_2 g^{(2)} + \dots + b_d g^{(d)}$. (Note: At this point all $g \in G$ are linear, and so are all the $g^{(k)}$.)
- 5 $S := \bigcup_{g \in G} \{g^{(1)}, g^{(2)}, \dots, g^{(d)}\}$
- 6 **return** $\ker_{\mathbb{Z}}(S)$

Note that the algorithm avoids the need of solving systems of diophantine equations by exploiting a priori knowledge on the structure of the solution set.

The case of non-irreducible polynomials is easily reduced to the former case by considering all pairs of factors. To be precise, let $p, q \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ be any polynomials, and let

$$p = p_1^{u_1} p_2^{u_2} \dots p_r^{u_r}, \quad q = q_1^{v_1} q_2^{v_2} \dots q_s^{v_s}$$

be their factorization into irreducible factors. Then

$$\text{Spread}(p, q) = \bigcup_{i=1}^r \bigcup_{j=1}^s \text{Spread}(p_i, q_j).$$

In short, given $p, q \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ we can compute $\text{Spread}(p, q)$ and therefore also $\text{Disp}_i(p, q)$ and $\text{Disp}(p, q)$.

Every given polynomial p can be split uniquely (up to constant multiples) into a factorization $p = uv$ where u is periodic and v is aperiodic. We call u and v the *periodic* and *aperiodic part* of p , respectively. As we can factor polynomials and compute, as described above, their spread, this decomposition can be computed.

EXAMPLE 2. 1. $p = n + k$, $q = n + 2k$. Here we have

$$p(n, k) - q(n + i, k + j) = (-i + 2j) - k$$

and $G = \{1\}$, hence $\text{Spread}(p, q) = \emptyset$.

2. $p = n + \sqrt{2}k$, $q = n + \sqrt{2}k + 3 - 2\sqrt{2}$. Here we have

$$p(n, k) - q(n + i, k + j) = -i - \sqrt{2}j - 3 + 2\sqrt{2}$$

and $G = \{-i - \sqrt{2}j - 3 + 2\sqrt{2}\}$. With $B = \{1, \sqrt{2}\}$ we get $S = \{-i - 3, -j + 2\}$, from which we obtain

$$\text{Spread}(p, q) = \{(-3, 2)\}.$$

3. $p = 3k^2 + 6kn - 7k + 3n^2 - 7n + 1$, $q = 3k^2 + 6kn - 13k + 3n^2 - 13n + 11$. Here we have

$$p(n, k) - q(n + i, k + j) = -(i + j - 1)(3i + 3j - 10) - 6(i + j - 1)k + 6(i + j - 1)n$$

and $G = \{i + j - 1\}$. With $B = \{1\}$, we get $S = G$ from which we obtain

$$\text{Spread}(p, q) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \mathbb{Z} \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

5. APERIODIC FACTORS IN DENOMINATORS OF SOLUTIONS

In this section we solve the following problem. **Given** a nonempty finite shift set $S \subseteq \mathbb{Z}^r$ with coefficients $a_s \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ ($\mathbf{s} \in S$), **find** a polynomial $d \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ such that for any solution $y = \frac{p}{uq}$ of (5) with $p, q, u \in \mathbb{K}[\mathbf{n}]$ where q is aperiodic and u is periodic, we have

$$q \mid d.$$

Such a d is called an *aperiodic universal denominator* (or *aperiodic denominator bound*) of (5).

First we generalize Theorem 1, i.e., for any solution $y \in \mathbb{K}(\mathbf{n})$ of (5) we bound the dispersion of the aperiodic denominator part of y . To be more precise, we first bound the dispersion w.r.t. one component n_i in $\mathbf{n}$.

LEMMA 2. *Let $S \subseteq \mathbb{Z}^r$ be finite and nonempty and let $a_s \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ for $\mathbf{s} \in S$, and $f \in \mathbb{K}[\mathbf{n}]$; let a'_s be the aperiodic part of a_s . Let $i \in \{1, \dots, r\}$. Define*

$$k := \max\{|a_i - b_i| : (a_1, \dots, a_r), (b_1, \dots, b_r) \in S\}$$

and let

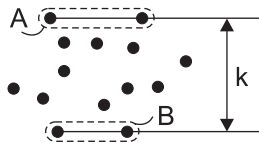
$$\begin{aligned} A &= \{(s_1, \dots, s_r) : (s_1, \dots, s_r) \in S \text{ and} \\ &\quad \exists (t_1, \dots, t_r) \in S \text{ s.t. } t_i - s_i = k\}, \\ B &= \{(s_1, \dots, s_r) : (s_1, \dots, s_r) \in S \text{ and} \\ &\quad \exists (t_1, \dots, t_r) \in S \text{ s.t. } s_i - t_i = k\}. \end{aligned}$$

Define

$$s_i := \max\{\text{Disp}_i(a'_s, N_i^{-k} a'_t) : \mathbf{s} \in A \text{ and } \mathbf{t} \in B\}.$$

Then for any solution $y = \frac{p}{uq} \in \mathbb{K}(\mathbf{n})$ of (5) with periodic part u and aperiodic part q we have

$$\text{Disp}_i(q) = \text{Disp}_i(q, q) \leq s_i.$$



PROOF. Suppose that $d := \text{Disp}_i(q) > s_i$. Then we find irreducible factors u and v of q such that

$$N^{\mathbf{e}}u = v \quad (6)$$

for some $\mathbf{e} = (e_1, \dots, e_r) \in \mathbb{Z}^r$ with $e_i = d$. Consider all the factors $N^{\mathbf{u}}u$ and $N^{\mathbf{v}}v$ occurring in q where the i th entries in $\mathbf{u}$ and $\mathbf{v}$ are 0. We choose now those factors from q where $\mathbf{u}$ and $\mathbf{v}$ are maximal w.r.t. lexicographic order; these factors are denoted by u' and v' , respectively.

First suppose that u' divides one of the polynomials a_s with $\mathbf{s} \in A$. As S is not empty, B is not empty. Therefore, we can choose that polynomial $a_{\mathbf{w}}$ with $\mathbf{w} = (w_1, \dots, w_r) \in B$ such that $(w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_r)$ is maximal w.r.t. lexicographic order. By (5) we can write

$$N^{\mathbf{w}}y = \frac{1}{a_{\mathbf{w}}} \left(f - \sum_{\mathbf{s} \in S \setminus \{\mathbf{w}\}} a_{\mathbf{s}} N^{\mathbf{s}}y \right). \quad (7)$$

Now observe that the factor $N^{\mathbf{w}}v'$ does not occur in the denominator of any $N^{\mathbf{s}}y$ with $\mathbf{s} \in S \setminus \{\mathbf{w}\}$: if $N^{\mathbf{w}}v'$ occurred in such a denominator, $\mathbf{s} \in B$ by construction (recall that u' and v' have maximal distance d in the i -coordinate among all factors in q and that v' with $N^{\mathbf{w}}v'$ is shifted maximally by k among all possible choices from S in direction of the i -coordinate since $\mathbf{w} \in B$; so only if $\mathbf{s} \in B$ is necessary to guarantee that $N^{\mathbf{w}}v'$ is a factor of $N^{\mathbf{s}}q$ and hence of the denominator of $N^{\mathbf{s}}y$). But then by the assumption that $(w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_r)$ is maximal w.r.t. lexicographic order (among all possible choices) and that $v' = N^{\mathbf{v}}v$ with $\mathbf{v} = (v_1, \dots, v_{i-1}, 0, v_{i+1}, \dots, v_r)$ is maximal w.r.t. lexicographical order, it follows that $N^{\mathbf{w}}v'$ can only occur in the denominator of $N^{\mathbf{w}}y$. Summarizing, the factor $N^{\mathbf{w}}v'$ does not occur in the denominators of $N^{\mathbf{s}}y$ for any $\mathbf{s} \in S \setminus \{\mathbf{w}\}$, and since $f, a_s \in \mathbb{K}[\mathbf{n}]$, the common denominator of $f - \sum_{\mathbf{s} \in S \setminus \{\mathbf{w}\}} a_{\mathbf{s}} N^{\mathbf{s}}y$ does not contain the factor $N^{\mathbf{w}}v'$. Moreover, since u' is a factor of a_s for some $\mathbf{s} \in A$, and since $\mathbf{w} \in B$ and $d > s_i$, $N^{\mathbf{w}}v'$ cannot be a factor of a_s for any $\mathbf{s} \in B$. In particular, our $a_{\mathbf{w}}$ is not divisible by $N^{\mathbf{w}}v'$. Overall, the common denominator of the right hand side of (7) cannot contain the factor $N^{\mathbf{w}}v'$ which implies that the denominator of $N^{\mathbf{w}}y$ is not divisible by $N^{\mathbf{w}}v'$. Thus the denominator of y , in particular q is not divisible by v' ; a contradiction.

Conversely, suppose that u' does not divide any of the polynomials a_s with $\mathbf{s} \in A$. Then by similar arguments as above (the roles of A and B exchanged), we derive again a contradiction. Therefore $s_i \leq \text{Disp}_i(q)$. $\square$

EXAMPLE 3. *In the generic univariate case (1) ($r = i = 1$) the shift set is $S = \{0, 1, \dots, m\} \subseteq \mathbb{Z}^1$ and for the sets A and B from Lemma 2 we have $A = \{0\}$ and $B = \{m\}$. In this particular instance, Lemma 2 corresponds to Theorem 1.*

A bound of the dispersion for the multivariate case is given in the next theorem.

THEOREM 2. *Let $S \subseteq \mathbb{Z}^r$ be finite and nonempty and let $a_s \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ for $\mathbf{s} \in S$, and $f \in \mathbb{K}[\mathbf{n}]$. Then one can compute an $s \in \mathbb{N} \cup \{-\infty\}$ with the following property: For any solution $y = \frac{p}{uq} \in \mathbb{K}(\mathbf{n})$ of (5) with periodic part u and aperiodic part q we have*

$$\text{Disp}(q) = \text{Disp}(q, q) \leq s. \quad (8)$$

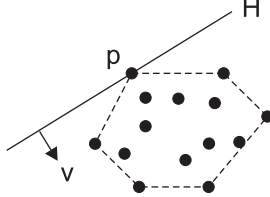
PROOF. Compute the values s_i for $i \in \{1, \dots, r\}$ as described in Lemma 2; the spread can be computed with Algorithm 1. By taking $s = \max(s_1, \dots, s_r)$ the property $\text{Disp}(q) = \text{Disp}(q, q) \leq s$ is guaranteed. $\square$

In order to derive an aperiodic denominator bound, we adapt the idea presented in Section 2. Namely, we will choose an appropriate point $\mathbf{p} \in S$ and express $N^{\mathbf{p}}y$ for any solution $y \in \mathbb{K}(\mathbf{n})$ of (5) in terms of $N^{\mathbf{s}}y$ for points $\mathbf{s} \in S'$ which are sufficiently far away from $\mathbf{p}$. To be more precise, for any $s > 0$ we can explicitly write

$$N^{\mathbf{p}}y = \frac{b + \sum_{\mathbf{i} \in S'} b_{\mathbf{i}} N^{\mathbf{i}}y}{\prod_{\mathbf{i} \in W - \mathbf{p}} N^{\mathbf{i}}a_{\mathbf{p}}} \quad (9)$$

for some polynomials $b, b_{\mathbf{i}} \in \mathbb{K}[\mathbf{n}]$ and for finite sets $W, S' \subseteq \mathbb{Z}^r$ with the following property: the distance of the points S' to $\mathbf{p}$ is at least s . Then by taking s as in Theorem 2, we will be able to conclude that $\prod_{\mathbf{i} \in W - \mathbf{p}} N^{\mathbf{i}}a_{\mathbf{p}}$ is an aperiodic denominator bound of $N^{\mathbf{p}}y$, and consequently $\prod_{\mathbf{i} \in W} N^{\mathbf{i}-2\mathbf{p}}a_{\mathbf{p}}$ is an aperiodic denominator bound of y .

Such appropriate points $\mathbf{p}$ from S can be chosen as follows. Let $S \subseteq \mathbb{Z}^r$ be a finite set. A point $\mathbf{p} \in S$ is called a *corner point* (or *extreme point*) of S , if there exists an affine hyperplane H (codimension 1) which contains $\mathbf{p}$ and where all other points $S \setminus \{\mathbf{p}\}$ are situated in one of the two open half spaces determined by the hyperplane.



Such an affine hyperplane H is called *border plane* of S for $\mathbf{p}$, and a vector being orthogonal to H and directing to the half space of the points $S \setminus \{\mathbf{p}\}$ is called *inner vector*.

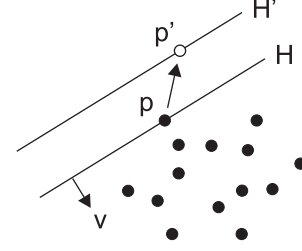
Note that the corner points are the extreme points of the convex hull generated by S , and they can be computed by simple linear algebra; for further details see, e.g., [10].

EXAMPLE 4. In the generic univariate case (1) ($r = 1$ and $S = \{0, 1, \dots, m\}$) the corner points are 0 and m , and the border planes are $\{0\}$ and $\{m\}$ with inner vectors 1 and -1 , respectively.

More generally, if we are given a finite set $S \subseteq \mathbb{Z}^r$ with $(0, \dots, 0) \in S$ and $\max\{d_i : (d_1, \dots, d_r) \in S\} > 0$ for each $1 \leq i \leq r$, there are at least $r + 1$ corner points.

For our denominator bound construction we start with the following simple lemma.

LEMMA 3. Let $S \subseteq \mathbb{Z}^r$ be a nonempty finite set and let $\mathbf{p} \in S$ be a corner point together with a border plane H and inner vector $\mathbf{v}$. Consider any hyperplane H' which is parallel to H . Then for any $\mathbf{p}' \in H' \cap \mathbb{Z}^r$ the points $S + (\mathbf{p}' - \mathbf{p}) \setminus \{\mathbf{p}'\}$ are all outside of H' in the half space determined by the direction of $\mathbf{v}$.



PROOF. Since $H + (\mathbf{p}' - \mathbf{p}) = H'$, H' is a border plane of $S + (\mathbf{p}' - \mathbf{p})$ for $\mathbf{p}'$. This proves the lemma. $\square$

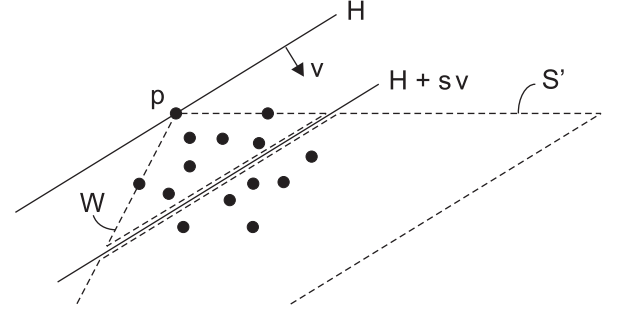
By iterative application of the previous lemma we obtain the following theorem.

THEOREM 3. Let $S \subseteq \mathbb{Z}^r$ be nonempty and finite, and let $a_{\mathbf{s}} \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ for $\mathbf{s} \in S$, and $f \in \mathbb{K}[\mathbf{n}]$. Let $\mathbf{p}$ be a corner point of S with a border plane H and inner vector $\mathbf{v}$. Then for every $s > 0$ there exist finite sets

$$W \subseteq \mathbb{Z}^r \cap \bigcup_{0 \leq e \leq s} (H + e\mathbf{v}) \text{ and} \quad (10)$$

$$S' \subseteq \mathbb{Z}^r \cap \bigcup_{e > 0} (H + (s + e)\mathbf{v}), \quad (11)$$

and polynomials $b, b_{\mathbf{i}} \in \mathbb{K}[\mathbf{n}]$ such that for any solution $y \in \mathbb{K}(\mathbf{n})$ of (5) the relation (9) holds.



PROOF. We show the theorem for a generic solution $y \in \mathbb{K}(\mathbf{n})$ of (5). Hence the ingredients $b, b_{\mathbf{i}} \in \mathbb{K}[\mathbf{n}]$ and W, S' under consideration will hold for any specific solution as stated in the theorem. For $\tilde{S} := S \setminus \{\mathbf{p}\}$ we have

$$N^{\mathbf{p}}y = \frac{f - \sum_{\mathbf{s} \in \tilde{S}} a_{\mathbf{s}} N^{\mathbf{s}}y}{a_{\mathbf{p}}}$$

by (5). If $\tilde{S} = \{\}$, take $S' = \{\}$ and $W = \{\mathbf{p}\}$ and we are done. Otherwise, let $\mathbf{p}' \in \tilde{S}$ be such that the distance to H is minimal. Define $H' := H + (\mathbf{p}' - \mathbf{p})$. If $H' \subseteq \{H + (s + e)\mathbf{v} : e > 0\}$, we are again done with $W = \{\mathbf{p}\}$ and $S' = S \setminus \{\mathbf{p}\}$. Now suppose that $H' \subseteq \{H + e\mathbf{v} : 0 \leq e \leq s\}$, and let $\{\mathbf{p}_1, \dots, \mathbf{p}_k\} = H' \cap \tilde{S}$ (by construction $\mathbf{p}' \in H' \cap \tilde{S}$) and define $S_1 = \tilde{S} \setminus \{\mathbf{p}_1, \dots, \mathbf{p}_k\}$. Then we can write

$$N^{\mathbf{p}}y = \frac{f - \sum_{\mathbf{s} \in S_1} a_{\mathbf{s}} N^{\mathbf{s}}y - \sum_{\mathbf{s} \in \{\mathbf{p}_1, \dots, \mathbf{p}_k\}} a_{\mathbf{s}} N^{\mathbf{s}}y}{a_{\mathbf{p}}}. \quad (12)$$

By (5)

$$N^{\mathbf{p}_1}y = \frac{N^{\mathbf{p}_1 - \mathbf{p}}f - \sum_{\mathbf{s} \in \tilde{S} + (\mathbf{p}_1 - \mathbf{p})} a_{\mathbf{s}} N^{\mathbf{s}}y}{N^{\mathbf{p}_1 - \mathbf{p}}a_{\mathbf{p}}}. \quad (13)$$

In particular, by Lemma 3 the points from $\tilde{S} + (\mathbf{p}_1 - \mathbf{p})$ are all outside of H' in the half space determined by the direction

of $\mathbf{v}$. Thus we substitute (13) into (12) and can express $N^{\mathbf{p}}y$ in the form

$$N^{\mathbf{p}}y = \frac{f' - \sum_{\mathbf{s} \in S'_1} a'_s N^{\mathbf{s}}y - \sum_{\mathbf{s} \in \{\mathbf{p}_2, \dots, \mathbf{p}_k\}} a_s N^{\mathbf{s}}y}{a_{\mathbf{p}} N^{\mathbf{p}_1 - \mathbf{p}} a_{\mathbf{p}}}$$

for some $f' \in \mathbb{K}[\mathbf{n}]$ and $a'_s \in \mathbb{K}[\mathbf{n}]$ with $S'_1 = S_1 \cup (\tilde{S} + \mathbf{p}_1 - \mathbf{p})$. After $k - 1$ further reductions, we end up at the form

$$N^{\mathbf{p}}y = \frac{b - \sum_{\mathbf{s} \in S''_1} b_s N^{\mathbf{s}}y}{\prod_{\mathbf{i} \in W_1 - \mathbf{p}} N^{\mathbf{i}} a_{\mathbf{p}}}$$

for some $b, b_s \in \mathbb{K}[\mathbf{n}]$ and with $W_1 = \{\mathbf{p}, \mathbf{p}_1, \dots, \mathbf{p}_k\}$ and $S''_1 = S_1 \cup \bigcup_{1 \leq i \leq k} \tilde{S} + (\mathbf{p}_i - \mathbf{p})$. Note that all points $\mathbf{p}_1, \dots, \mathbf{p}_k$ which are closest to H have been eliminated. Now we repeat the construction from above until we enter in the base case given in the beginning of the proof. This completes the proof. $\square$

Note that all the ingredients $W, S', b \in \mathbb{K}[\mathbf{n}]$ and the b_s for $\mathbf{s} \in W$ can be computed explicitly. However, for getting an aperiodic denominator bound, we only need W . The proof of Theorem 3 delivers the following simple algorithm.

ALGORITHM 2. *Input:* A finite nonempty set $S \subseteq \mathbb{Z}^r$ and a corner point $\mathbf{p}$ of S with a border plane H and inner vector $\mathbf{v}$; $s > 0$.

Output: A finite set $W \subseteq \mathbb{Z}^r$ with (10) such that there are S' with (11) and $b, b_i \in \mathbb{K}[\mathbf{n}]$ such that (9) holds for any solution $y \in \mathbb{K}(\mathbf{n})$ of (5).

```

1   $\tilde{S} := S \setminus \{\mathbf{p}\}; S' := S \setminus \{\mathbf{p}\}; W := \{\mathbf{p}\}$ 
2  while  $\bigcup_{0 \leq c \leq s} (H + c\mathbf{v}) \cap S' \neq \{\}$  do
3    Let  $\{\mathbf{p}_1, \dots, \mathbf{p}_k\}$  be the points in  $S'$  which have minimal distance to  $H$ .
4     $W := W \cup \{\mathbf{p}_1, \dots, \mathbf{p}_k\}$ 
5     $S' := S' \setminus \{\mathbf{p}_1, \dots, \mathbf{p}_k\}$ 
6     $S' := S' \cup \bigcup_{1 \leq i \leq k} \tilde{S} + (\mathbf{p}_i - \mathbf{p})$ 
7  enddo
8  return  $W$ 
```

Finally, we end up at the following theorem which tells us how we can compute an aperiodic denominator bound.

THEOREM 4. *Let $S \subseteq \mathbb{Z}^r$ be finite and nonempty, and let $a_s \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ for $\mathbf{s} \in S$, and $f \in \mathbb{K}[\mathbf{n}]$; let a'_s be the aperiodic part of a_s .*

Take $s \in \mathbb{N} \cup \{-\infty\}$ s.t. for any solution $y = \frac{p}{uq} \in \mathbb{K}(\mathbf{n})$ of (5) with periodic part u and aperiodic part q we have (8). Let $\mathbf{p}$ be a corner point of S with a border plane H and inner vector $\mathbf{v}$ with $|\mathbf{v}| \geq 1$. Let W be the output of Algorithm 2 with input $H, \mathbf{p}$ and s . Then

$$\prod_{\mathbf{s} \in W - 2\mathbf{p}} N^{\mathbf{s}} a'_{\mathbf{p}}$$

is an aperiodic universal denominator of (5).

PROOF. Let $y = \frac{p}{uq} \in \mathbb{K}(\mathbf{n})$ be a solution of (5) with periodic part u and aperiodic part q . By construction of W it follows that there are S' and $b, b_i \in \mathbb{K}[\mathbf{n}]$ with (11) and (9). Since $|\mathbf{v}| \geq 1$, the distance of the points S' to $\mathbf{p}$ is larger than s . Observe that $N^{\mathbf{p}}q$ must occur in the denominator of the right hand side of (9). Using (8), the aperiodic denominator

parts in $N^{\mathbf{s}}q$ with $\mathbf{s} \in S'$ cannot contribute to the aperiodic denominator part of $N^{\mathbf{p}}q$. Hence only the polynomial $\prod_{\mathbf{s} \in W - \mathbf{p}} N^{\mathbf{s}} a_{\mathbf{p}}$ is responsible for $N^{\mathbf{p}}q$, i.e.,

$$N^{\mathbf{p}}q \mid \prod_{\mathbf{s} \in W - \mathbf{p}} N^{\mathbf{s}} a_{\mathbf{p}}.$$

Thus $\prod_{\mathbf{s} \in W - 2\mathbf{p}} N^{\mathbf{s}} a_{\mathbf{p}}$ is a universal denominator of (5). $\square$

EXAMPLE 5. *In the generic univariate case (1) ($r = 1$ and $S = \{0, \dots, m\}$) the coefficients are given by a_i for $0 \leq i \leq m$. First, we take the corner point m with the border plane $H = \{m\}$ and inner vector -1 . Note that with (3) it follows that (8) holds for any solution $y = \frac{p}{q} \in \mathbb{K}(\mathbf{n})$; see also Example 3. Applying Algorithm 2 with the input $S, m, H, -1$, we get $W = \{m, m-1, \dots, m-s\}$. Hence Theorem 4 delivers the universal denominator bound (here we have only aperiodic factors)*

$$\prod_{\mathbf{s} \in W - 2m} N^{\mathbf{s}} a_m$$

which agrees with (4). Similarly, if we take the corner point 0 with the inner vector 1, we get $W = \{0, 1, \dots, s\}$ and obtain the universal denominator bound

$$\prod_{\mathbf{s} \in W - 2 \cdot 0} N^{\mathbf{s}} a_0 = \prod_{i=0}^s N^i a_0.$$

Combining these two estimates produces (2).

From the point of view of application the following remarks are in place.

1. $s \in \mathbb{N} \cup \{-\infty\}$ can be computed by Theorem 2 and by applying Algorithm 1.
2. Applying Algorithm 2 we can compute the finite set $W \subseteq \mathbb{Z}^r$; here we remark that different choices of the border plane H might lead to sets W of different size. Exploiting the particular structure of S gives room for improvement.
3. Suppose that we are given k corner points with corresponding border planes. Then by Theorem 4 we end up at different aperiodic universal denominators, say $d_1, \dots, d_k \in \mathbb{K}[\mathbf{n}]$. Then taking

$$\gcd(d_1, \dots, d_k)$$

leads to a sharper universal bound.

4. The coefficients a_s with $\mathbf{s} \in S$ are often available in factorized form. Then also the d_i s are obtained in factorized form, and the gcd-computations boil down to comparisons of these factors and bookkeeping of their multiplicities.

Combining the aperiodic denominator bounds for different corner points gives the following result.

THEOREM 5. *Let $S \subseteq \mathbb{Z}^r$ be finite and nonempty and let $a_s \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ for $\mathbf{s} \in S$, and $f \in \mathbb{K}[\mathbf{n}]$. Let $\mathbf{p}_1, \dots, \mathbf{p}_k \in S$ be corner points of S . If the denominator of a rational solution of (5) contains an aperiodic irreducible factor, then shift equivalent factors occur in each of the coefficients $a_{\mathbf{p}_1}, \dots, a_{\mathbf{p}_k}$.*

PROOF. By Theorem 4 aperiodic denominator bounds can be derived by the corner points $\mathbf{p}_j$ in the form $\prod_i N^{i_{\mathbf{p}_j}}$, respectively. Hence an aperiodic denominator bound of (5) can be written in the form

$$d = \gcd\left(\prod_i N^{i_{\mathbf{p}_1}}, \dots, \prod_i N^{i_{\mathbf{p}_k}}\right).$$

If a rational solution contains an aperiodic irreducible factor h , then h is also contained in d . Hence h or a shift equivalent factor occurs in each of the $a_{\mathbf{p}_1}, \dots, a_{\mathbf{p}_k}$. $\square$

The following special cases are immediate.

COROLLARY 1. Let $S \subseteq \mathbb{Z}^r$ be finite and nonempty, and let $a_{\mathbf{s}} \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ for $\mathbf{s} \in S$, and $f \in \mathbb{K}[\mathbf{n}]$. Let $\mathbf{s}, \mathbf{t} \in S$ be two corner points and let $a'_{\mathbf{s}}$ and $a'_{\mathbf{t}}$ be the aperiodic parts of the coefficients $a_{\mathbf{s}}$ and $a_{\mathbf{t}}$, respectively. If $\text{Disp}(a'_{\mathbf{s}}, a'_{\mathbf{t}}) = -\infty$, then the aperiodic denominator part of any rational solution of (5) is 1.

COROLLARY 2. Let $S \subseteq \mathbb{Z}^r$ be finite and nonempty, and let $a_{\mathbf{s}} \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ for $\mathbf{s} \in S$, and $f \in \mathbb{K}[\mathbf{n}]$. If there is a corner point of S whose coefficient has no aperiodic factor, then the aperiodic denominator part of any rational solution of (5) is 1.

Besides these structural consequences, Theorem 5 provides the following improvement of our aperiodic denominator bound algorithm. To be more precise, Lemma 2 and thus Theorem 2 can be improved in the following way. In the proof of Lemma 2 we assume that there are irreducible factors u and v in the denominator of the solution $y \in \mathbb{K}(\mathbf{n})$ of (5) such that (6) for some $\mathbf{e} = (e_1, \dots, e_r) \in \mathbb{Z}^r$ with $e_i = d$ where d is larger than s_i . By the choice of s_i this leads to a contradiction. Now we exploit in addition Theorem 5: the factors u and v can be only factors that occur –up to shift equivalence– in each coefficient of the corner points $\mathbf{p}_1, \dots, \mathbf{p}_k$. Hence it suffices to choose s_i as summarized in the following proposition.

PROPOSITION 1. Let $S \subseteq \mathbb{Z}^r$ be finite and nonempty, and let $a_{\mathbf{s}} \in \mathbb{K}[\mathbf{n}] \setminus \{0\}$ for $\mathbf{s} \in S$, and $f \in \mathbb{K}[\mathbf{n}]$. Let $\mathbf{p}_1, \dots, \mathbf{p}_k$ be corner points of S , and let $a'_{\mathbf{s}}$ be the aperiodic part of $a_{\mathbf{s}}$ whose factors are present –up to shift equivalence– in each coefficient of the corner points. Let $i \in \{1, \dots, r\}$. Define

$$k := \max\{|a_i - b_i| : (a_1, \dots, a_r), (b_1, \dots, b_r) \in S\}$$

and let

$$\begin{aligned} A &= \{(s_1, \dots, s_r) : (s_1, \dots, s_r) \in S \text{ and} \\ &\quad \exists (t_1, \dots, t_r) \in S \text{ s.t. } t_i - s_i = k\} \\ B &= \{(s_1, \dots, s_r) : (s_1, \dots, s_r) \in S \text{ and} \\ &\quad \exists (t_1, \dots, t_r) \in S \text{ s.t. } s_i - t_i = k\}. \end{aligned}$$

Define

$$s_i := \max\{\text{Disp}_i(a'_{\mathbf{s}}, N_i^{-k} a'_{\mathbf{t}}) : \mathbf{s} \in A \text{ and } \mathbf{t} \in B\}.$$

Then for any solution $y = \frac{p}{uq} \in \mathbb{K}(\mathbf{n})$ of (5) with periodic part u and aperiodic part q we have

$$\text{Disp}_i(q) = \text{Disp}_i(q, q) \leq s_i.$$

6. EXAMPLES

EXAMPLE 6. Consider the recurrence

$$\begin{aligned} &(2kn+1)(6k^2+12k-4n^2-4n+5)f(n, k) \\ &+ (2kn+4k+1)(6k^2+10k+4n^2+8n-7)f(n+1, k) \\ &- (2kn+8n+1)(6k^2+24k+4n^2-20n-7)f(n, k+2) \\ &- (2kn+4k+8n+17) \\ &\quad \times (6k^2+22k-4n^2+16n+45)f(n+1, k+2) = 0. \end{aligned}$$

The maximum spread among the coefficients of this recurrence is $s = 4$.

Every point in the shift set $\{(0, 0), (1, 0), (0, 2), (1, 2)\}$ qualifies as a corner point. We choose $\mathbf{p} = (0, 0)$ as corner point and let H be the plane through $\mathbf{p}$ orthogonal to $\mathbf{v} = (1, 1)$.

Algorithm 2 delivers the set

$$\begin{aligned} &\{(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0), (8, 0), \\ &(0, 2), (1, 2), (2, 2), (3, 2), (4, 2), (5, 2), (6, 2) \\ &(0, 4), (1, 4), (2, 4), (3, 4), (4, 4), \\ &(0, 6), (1, 6), (2, 6), \\ &(0, 8)\} \end{aligned}$$

as W , from which by Theorem 4 it follows that

$$\prod_{i=0}^8 \prod_{j=0}^{4-\lceil i/2 \rceil} N^i K^{2j} (2kn+1)(6k^2+12k-4n^2-4n+5)$$

is a universal aperiodic denominator.

Taking instead $(1, 2)$ as corner point gives the aperiodic denominator bound

$$\begin{aligned} &\prod_{i=-8}^0 \prod_{j=-4-\lceil i/2 \rceil}^0 N^{i-1} K^{2j-2} ((2kn+4k+8n+17) \\ &\quad \times (6k^2+22k-4n^2+16n+45)). \end{aligned}$$

The greatest common divisor of the two polynomials is

$$(2kn+1)(2(k+2)n+1)(2k(n+1)+1)(2(k+2)(n+1)+1).$$

This is exactly the denominator of the actual solution

$$\frac{3k+n}{(2kn+1)(2(k+2)n+1)(2k(n+1)+1)(2(k+2)(n+1)+1)}$$

of the recurrence.

The computation could have been simplified by disregarding the factors $p = 6k^2+12k-4n^2-4n+5$ and $q = 6k^2+22k-4n^2+16n+45$. Because of

$$\text{Spread}(p, q) = \{ \},$$

they cannot contribute to the universal denominator (compare Theorem 5).

EXAMPLE 7. Some corner points may be easier to handle than others. As an example, consider the equation

$$\begin{aligned} &(k^2+n^2+1)(2k^4+4k^3+4k^2n^2+4k^2n+6k^2+4kn^2 \\ &+ 8kn+9k+2n^4+4n^3+6n^2+3n+4)f(n, k) \\ &- (k^2+4k+n^2+5)(2k^4+4k^3+4k^2n^2+4k^2n+6k^2-1 \\ &+ 4kn^2-2kn-k+2n^4+4n^3+6n^2-2n)f(n, k+1) \\ &- (2k+1)(n+1)(k^2+n^2+4n+5)f(n+1, k) = 0. \end{aligned}$$

Without any computation, it can be deduced that any potential aperiodic factor in a denominator must be a shifted copy

of $k^2 + n^2 + 4n + 5 = (n + 2)^2 + k^2 + 1$. Indeed, a rational solution of the equation is given by

$$\frac{k^2 + n^2}{(k^2 + n^2 + 1)((k + 1)^2 + n^2 + 1)(k^2 + (n + 1)^2 + 1)}.$$

EXAMPLE 8. Theorem 4 is not sufficient for predicting periodic factors of a denominator. As an example, consider the equation

$$\begin{aligned} &2(k + n + 1)f(n, k) - (k + 3n + 8)f(n, k + 1) \\ &- (5k + 3n + 12)f(n + 1, k) + 3(k + n + 5)f(n + 1, k + 1) \\ &+ (k + n + 5)f(n + 2, k) = 0. \end{aligned}$$

Possible choices for corner points are $(0, 0), (0, 1), (1, 1)$ and $(2, 0)$. Because of

$$\text{Spread}(k + n + 1, 3n + k + 8) = \{\},$$

one might be tempted to believe that only trivial denominators can occur in a solution. However, the equation admits the nontrivial rational function solution

$$\frac{n^2 + k^2}{(k + n + 1)(k + n + 2)(k + n + 3)}.$$

Observe that although not **every** corner point contains a shifted copy of $k + n + 1$, there is still **some** corner point which does.

This need not be the case, as indicated by the example

$$f(n + 1, k) - f(n, k + 1) = 0$$

already mentioned in the introduction. This example, however, is special because the shift set $\{(0, 1), (1, 0)\}$ belongs to a proper affine subspace of $\mathbb{Z}^r$, and whenever this is the case, say the shift set belongs to a subspace $L \subsetneq \mathbb{Z}^r$, then for every vector $v = (v_1, \dots, v_r) \in L^\perp \setminus \{0\}$ the polynomial $p = v_1 n_1 + \dots + v_r n_r$ has the property that f is a rational solution of the equation if and only if $f p^\alpha$ is, for any $\alpha \in \mathbb{Z}$. In particular, if there exists a rational solution at all, then there also exists one whose denominator does not contain p .

Apart from this exceptional situation, we observed on all the examples we considered that periodic factors of a denominator appeared (possibly as a shifted copy) in at least one of the coefficients corresponding to some corner point of the shift set. This at least suggests the periodic factors of these coefficients as plausible guesses for the periodic part of the denominator bound.

7. CONCLUSION

There are polynomials in several variables which may have an infinite spread. Such polynomials are called periodic, while polynomials that cannot have infinite spread are called aperiodic. Each polynomial can be split into a periodic and an aperiodic part.

We have shown that for partial linear difference equations with polynomial coefficients it is possible to determine all the factors that may possibly occur in the aperiodic part of the denominator of a rational function solution. The construction is a generalization of the corresponding result for univariate equations. It probably admits further generalization to q -equations or equations whose coefficients belong to a $\Pi\Sigma$ -field. As it stands, Theorem 4 will tend to produce only a rough bound for the aperiodic part of the denominator, but we have pointed out several refinements for improving the efficiency of the computation on concrete examples.

Eventually, it would be interesting to see whether it is possible to come up with an Abramov-style algorithm for directly computing the greatest common divisor of all the individual bounds obtained from each corner point. Until now, we have not succeeded in constructing such an algorithm.

It also remains open how to bound periodic factors of the denominator. The situation illustrated in Example 8, which seems to be typical both for random equations and for equations coming from applications, indicates that our result for aperiodic factors does not directly extend to periodic ones. On the other hand, it also indicates that an equation typically provides some hints for the periodic factors in the denominators of its rational solutions. This is useful for making plausible heuristic guesses. It also gives some hope that at least for certain types of equations the periodic part of a denominator can be found algorithmically. This needs further investigation.

8. REFERENCES

- [1] Sergei A. Abramov. On the summation of rational functions. *Zh. vychisl. mat. Fiz.*, pages 1071–1075, 1971.
- [2] Sergei A. Abramov. Problems in computer algebra that are connected with a search for polynomial solutions of linear differential and difference equations. *Moscow Univ. Comput. Math. Cybernet.*, 3:63–68, 1989.
- [3] Sergei A. Abramov. Rational solutions of linear difference and q -difference equations with polynomial coefficients. In *Proc. ISSAC'95*, July 1995.
- [4] Sergei A. Abramov and K.Yu. Kvashenko. Fast algorithms to search for the rational solutions of linear differential equations with polynomial coefficients. In *Proc. ISSAC'91*, pages 267–270, 1991.
- [5] Moulay Barkatou. Rational solutions of matrix difference equations: The problem of equivalence and factorization. In *Proc. ISSAC'99*, pages 277–282, 1999.
- [6] Alin Bostan, Frederic Chyzak, Thomas Cluzeau, and Bruno Salvy. Low complexity algorithms for linear recurrences. In Jean-Guillaume Dumas, editor, *Proc. ISSAC'06*, pages 31–39, 2006.
- [7] Manuel Bronstein. On solutions of linear ordinary difference equations in their coefficient field. *Journal of Symbolic Computation*, 29:841–877, 2000.
- [8] William Y. C. Chen, Peter Paule, and Husam L. Saad. Converging to Gosper's algorithm. *Adv. in Appl. Math.*, 41(3):351–364, 2008.
- [9] Amel Gheffar and Sergei Abramov. Valuations of rational solutions of linear difference equations at irreducible polynomials. (*submitted*).
- [10] Serge Lang. *Linear Algebra*. Springer, 1987.
- [11] Marko Petkovšek. Hypergeometric solutions of linear recurrences with polynomial coefficients. *Journal of Symbolic Computation*, 14(2-3):243–264, 1992.
- [12] Carsten Schneider. A collection of denominator bounds to solve parameterized linear difference equations in $\Pi\Sigma$ -extensions. In *Proc. SYNASC'04*, pages 269–282, 2004.
- [13] Mark van Hoeij. Rational solutions of linear difference equations. In *Proc. ISSAC'98*, pages 120–123, 1998.
- [14] Christian Weixlbaumer. Solutions of difference equations with polynomial coefficients. Master's thesis, RISC-Linz, 2001.

Real and Complex Polynomial Root-Finding with Eigen-Solving and Preprocessing

Victor Y. Pan

Department of Math and Computer Science
Lehman College of CUNY
Bronx, NY 10468 USA
victor.pan@lehman.cuny.edu
<http://comet.lehman.cuny.edu/vpan/>

Ai-Long Zheng

Department of Mathematics
The Graduate Center of CUNY
New York, NY 10036 USA
azheng@yahoo.com

ABSTRACT

Recent progress on root-finding for polynomial and secular equations largely relied on eigen-solving for the associated companion and diagonal plus rank-one generalized companion matrices. By applying to them Rayleigh quotient iteration, we could have already competed with the current best polynomial root-finders, but we achieve further speedup by applying additive preprocessing. Moreover our novel rational maps of the input matrix enables us to direct the iteration to approximating only real roots, so that we dramatically accelerate their numerical computation in the important case where they are much less numerous than all complex roots.

Categories and Subject Descriptors

F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—*Computations on matrices, Computations on polynomials*; G.1.3 [Numerical Analysis]: Numerical Linear Algebra—*Eigenvalues and eigenvectors*

General Terms

Algorithms

Keywords

Polynomial Root-finding, Companion matrices, DPR1 matrices, Eigenvalues, Eigenvectors, Rayleigh quotients, Secular equation

1. INTRODUCTION

Matrix methods for univariate polynomial root-finding are increasingly popular. Matlab offers a root-finder that applies the QR eigen-solver to the companion matrix of the polynomial. Using the diagonal plus rank-one (hereafter DPR1) generalized companion matrices is an alternative, which also supports root-finding for the important secular equation (cf.

[7], [15]). Fortune's package EIGENSOLVE alternates the steps of the Weierstrass' (Durand–Kerner's) functional root-finding iteration and the QR algorithm applied to the DPR1 matrices [11], [16] and is competitive with the current best root-finder MPSOLVE by Bini and Fiorentino, based on the Börsch–Supan's (Aberth–Ehrlich's) iteration.

These methods rapidly converge globally (that is right from the start) according to ample empirical evidence, though with no formal support. The proof of fast global convergence is a hard open problem for these and almost all other known effective matrix eigen-solvers and polynomial root-finders. A sample exception is the divide-and-conquer root-finder in [21]. It is proved to support the record and nearly optimal asymptotic Boolean and arithmetic time for approximating all polynomial roots, but relies on some advanced sophisticated techniques and has never been implemented.

In [6] the Rayleigh quotient eigen-solving iteration, applied to the $n \times n$ companion or DPR1 matrices, approximates a root closest to a fixed complex value by using linear arithmetic time $c_{RQ}n$ per step (for a scalar c_{RQ}) and linear memory space. Even for approximating all roots (via deflation) the iteration competes with the Börsch–Supan's and Weierstrass' according to the tests in [6] (cf. also [1]).

The QR algorithm in [7] employs the input matrix structure to use linear memory space and only $c_{QRR}n$ arithmetic operations per step in the case of polynomials that have only real roots [24]. Papers [3] and [5] remove this restriction and support time bound $c_{QRC}n$, which is still linear, but with the constant c_{QRC} noticeably larger than c_{RQ} and even c_{QRR} .

The Rayleigh quotient iteration solves an ill conditioned (that is nearly singular) linear system of equations at every iteration steps near the solution, but with additive preprocessing from [28] we avoid this hurdle and moreover perform iteration steps faster preserving their convergence rate.

Based on Rayleigh–Ritz procedure we can approximate eigenspaces rather than just eigenvectors (see Remark 3.1). This would strengthen the divide-and-conquer option because the known techniques are more efficient for matrix deflation than for splitting polynomials into factors.

Our tests give upper hand to our variants of the Rayleigh quotient iteration versus repeated squaring of companion matrices, known to be inexpensive [10], [22] and in our Theorem 5.6 extended to the DPR1 matrices. We, however, combine squaring with other matrix functions to isolate the r real roots from the $n - r$ nonreal ones.

At this point we can deflate an $r \times r$ matrix having r real eigenvalues, equal to the r desired real roots, and then read-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

ily approximate them, but currently we focus on a variation where we do not deflate, but direct the Rayleigh quotient iteration towards the approximation of these real eigenvalues.

In both ways we accelerate by the factor $\frac{n}{r}$ the known algorithms. Indeed they compute all real roots not much faster than all complex roots in terms of both theoretical estimates [14] and the actual CPU time in MPSOLVE.

In the highly important applications to algebraic optimization only real roots are of interest and are typically much less numerous than the nonreal roots. In such cases our acceleration by the factor $\frac{n}{r}$ is dramatic.

We also show a matrix-free version of this root-finder, preserving its asymptotic arithmetic cost.

Our study can be of independent technical interest, e.g., we reviewed the additive preprocessing, applied it to eigen-solving for the companion and DPR1 matrices, simplified Rayleigh quotient iteration with Newton-like linearization and preprocessing, specialized it to the companion and DPR1 matrices, extended repeated squaring from companion to DPR1 matrices, and proposed novel matrix techniques for real root-finding.

We demonstrate the power of our algorithms with extensive tests, performed by the second author; otherwise the paper is due to the first author.

Because of the space limitation, we omit many details, proofs, and test results as well as the discussion on modifications, extensions and applications of our algorithms and on their combinations with each other and with the known root-finders and eigen-solvers. We leave this material to the TR and journal versions of the paper.

2. DEFINITIONS AND BASIC RESULTS

“Op” stands for “arithmetic operation”. M^T is the transpose of a matrix M , M^H is its Hermitian transpose.

$(M_1, \dots, M_k) = ((M_i^T)_{i=1}^k)^T$ is a $1 \times k$ block matrix with blocks $M_1, \dots, M_k$. $\text{diag}(M_1, \dots, M_k) = \text{diag}(M_i)_{i=1}^k$ is a $k \times k$ block diagonal matrix with diagonal blocks $M_1, \dots, M_k$.

$I = I_k = (\mathbf{e}_j)_{j=1}^k$ is the $k \times k$ identity matrix. $J = J_k = (\mathbf{e}_k, \dots, \mathbf{e}_1)$ is the $k \times k$ reflection matrix, $J^2 = I$.

$\|M\|$ is the 2-norm of a matrix M . Nonsingular matrices M and linear systems $M\mathbf{y} = \mathbf{f}$ are *ill conditioned* (that is close to singular) if the condition numbers $\text{cond}(M) = \|M\| \|M^{-1}\|$ are large (in the context of the task and environment), in which case the computation of the inverses M^{-1} and vectors $\mathbf{y}$ is prone to magnification of the input and rounding errors and requires a higher precision [12], [13]. Otherwise the matrix and the systems are *well conditioned*.

$\mathcal{R}(M) = \{\mathbf{z} : \mathbf{z} = M\mathbf{y}\}$ is the range of a matrix M . $\mathcal{N}(M) = \{\mathbf{x} : M\mathbf{x} = \mathbf{0}\}$ is its *null space*, made up of its *null vectors* $\mathbf{x}$. $\mathbb{S}$ is its right (resp. left) *invariant subspace* or *eigenspace* if $M\mathbb{S} \subseteq \mathbb{S}$ (resp. $\mathbb{S}M \subseteq \mathbb{S}$). $\{L, B\}$ and $\{L, C\}$ are its left and right *eigenpairs*, respectively, and $\{L, B, C\}$ is its *eigentriple* if B and C are matrices of full rank, $BM = LB$, and $MC = CL$. $\lambda = \lambda(M)$ is its eigenvalue, whereas $\mathbf{b}$ and $\mathbf{c}$ are the left and right associated eigenvectors if $L = \lambda$ is a scalar, $B = \mathbf{b}$ and $C = \mathbf{c}$ are vectors.

An eigenvalue $\lambda(M)$ has an algebraic multiplicity ν if it is a root of the multiplicity ν of the characteristic polynomial $\det(M - xI)$. The geometric multiplicity $\bar{\nu}$ of an eigenvalue is the dimension of the space of the right (as well as left) associated eigenvectors, $\bar{\nu} \leq \nu$. An eigenvalue is simple if $\bar{\nu} = \nu = 1$. Hereafter we list the n eigenvalues $\lambda_j = \lambda_j(M)$ for $j = 1, \dots, n$ (according to their algebraic multiplicities)

in the nonincreasing order, $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$.

$\Lambda_K(M)$ is the set $\{\lambda_j\}_{j \in K}$ for $K \subseteq \{1, 2, \dots, n\}$. $\Lambda(M) = \Lambda_K(M)$ for $K = \{1, 2, \dots, n\}$. $\mathbb{S}_K = \mathbb{S}(M, \Lambda_K)$ and $\mathbb{T}_K = \mathbb{T}(M, \Lambda_K)$ are the two eigenspaces of all left and right eigenvectors, respectively, associated with all eigenvalues in this set. The eigenspace $\mathbb{S}_{\{1, \dots, \nu\}}$ is *dominant* and the eigenspace $\mathbb{S}_{\{\nu+1, \dots, n\}}$ is *dominated* if $|\lambda_{\nu+1}/\lambda_\nu| < 1$.

“The SMW formula” is our abbreviation for the Sherman–Morrison–Woodbury inversion formula

$$M^{-1} = (K - UV^H)^{-1} = K^{-1} + K^{-1}UG^{-1}V^HK^{-1}, \quad (2.1)$$

where $M, K \in \mathbb{C}^{n \times n}$, $U, V \in \mathbb{C}^{n \times r}$, $0 < r < n$, $G = I_r - V^HK^{-1}U$, $K = M + UV^H$, and the matrices M and K are assumed to be nonsingular.

The required basic concepts and results on computations with structured (e.g., Cauchy, Toeplitz and Toeplitz-like) matrices can be found in [20] and the bibliography therein.

$$F_{\mathbf{p}} = \begin{pmatrix} 0 & & & -p_0 \\ 1 & \ddots & & -p_1 \\ & \ddots & \ddots & \vdots \\ & & \ddots & 0 & -p_{n-2} \\ & & & 1 & -p_{n-1} \end{pmatrix} \quad (2.2)$$

is the companion matrix of a monic polynomial $p(x) = x^n + \sum_{i=0}^{n-1} p_i x^i$, $\mathbf{p} = (p_i)_{i=0}^{n-1}$, having the coefficient vector $\mathbf{p}$.

$Z = F_{\mathbf{p}} + \mathbf{e}_n \mathbf{p}^T$ is the $n \times n$ downshift matrix.

LEMMA 2.1. *Given a scalar $a \neq 0$, a vector $\mathbf{w} = (w_i)_{i=0}^{n-1}$, and a nonsingular $n \times n$ diagonal matrix D , one can compute the vector $\mathbf{y} = (y_i)_{i=0}^{n-1} = (D + Z)^{-1} \mathbf{w}$ in at most $2n - 1$ ops.*

Here are our basic results on additive preprocessing (see [26, Theorem 3.1 and its corollaries]).

THEOREM 2.1. *Suppose M is an $n \times n$ matrix having a rank ρ and the nullity $\nu = n - \rho$, U and V are two matrices of size $n \times r$, and the matrix $K = M + UV^H$ is nonsingular. Then $r \geq \text{rank}(U) \geq \nu$, $\mathcal{N}(M) \subseteq \mathcal{R}(K^{-1}U)$. Furthermore $\mathcal{R}(K^{-1}U) = \mathcal{N}(M)$ if $\text{rank}(U) = \nu$.*

COROLLARY 2.1. *Under the assumptions of Theorem 2.1 (except for the equation $\text{rank}(U) = \nu$) we have*

- (a) $\mathcal{R}(K^{-1}UX) = \mathcal{N}(M)$ if $\mathcal{R}(X) = \mathcal{N}(MK^{-1}U)$,
- (b) the converse is true if $\text{rank}(K^{-1}U) = r$, and
- (c) $\mathcal{N}(MK^{-1}U) = \mathcal{N}(I_\nu - V^HK^{-1}U)$ if the matrix U has full rank.

In the papers [26] and [28] these results are applied to solving linear systems of equations and eigen-solving.

3. RQ/SQ AND NEWTON’S ITERATIONS

The *Rayleigh quotient* (hereafter *RQ*) iteration recursively updates approximate eigenpairs $\{\lambda^{(i)}, \mathbf{w}_i\}$,

$$\mathbf{y}_i = (M - \lambda^{(i)}I)^{-1} \mathbf{w}_i, \quad (3.1)$$

$$\lambda^{(i+1)} = \lambda^{(i)} + \frac{\mathbf{y}_i^H \mathbf{w}_i}{\mathbf{y}_i^H \mathbf{y}_i}, \quad (3.2)$$

$$c_i \approx 1/\sqrt{\mathbf{y}_i^H \mathbf{y}_i}, \quad \mathbf{w}_{i+1} = c_i \mathbf{y}_i \quad (3.3)$$

for $i = 0, 1, \dots$. It stops where

$$\|M\mathbf{w}_i - \lambda^{(i)}\mathbf{w}_i\| < \tau\|M\mathbf{w}_i\| \quad (3.4)$$

for a fixed tolerance τ . One can check this bound periodically and skip checking where $|\lambda^{(i+1)} - \lambda^{(i)}| > \tau|\lambda^{(i)}|$.

The iteration is essentially equivalent to computing Newton's update of an approximate eigenpair $\{\lambda^{(i-1)}, \mathbf{y}_{i-1}\}$ (see [31]). If $|\lambda^{(0)} - \lambda_g| \ll \min_{\{j:j \neq g\}} |\lambda^{(0)} - \lambda_j|$, then $\lambda^{(i)} \rightarrow \lambda_g$ with quadratic rate [30].

The *Subspace and Rayleigh–Ritz* iterations converge under weaker assumptions [4], [30] and spilt out the eigenspaces associated with a fixed number of eigenvalues, thus supporting divide-and-conquer option, which is highly promising but requires further elaboration.

According to both formal and empirical study, the RQ iteration remains effective where the pairs $\{\lambda^{(i)}, \mathbf{y}_i\}$ are replaced with approximations $\{\tilde{\lambda}^{(i)}, \tilde{\mathbf{y}}_i\}$ such that $(\lambda_g - \tilde{\lambda}^{(i)})^{-1} \gg (\lambda_j - \tilde{\lambda}^{(i)})^{-1}$ for all j not equal to g . To save ops we can replace the ratios c_i in (3.3) with crude approximations and replace the RQs in (3.2) with *simple quotients* or *SQs*,

$$\lambda^{(i+1)} = \lambda^{(i)} + \frac{\mathbf{e}_j^H \mathbf{w}_i}{\mathbf{e}_j^H \mathbf{y}_i}, \quad \mathbf{e}_j^H \mathbf{y}_i \neq 0. \quad (3.5)$$

As $\lambda^{(i)} \rightarrow \lambda_j(M)$ we have $\mathbf{w}_i \rightarrow \mathcal{N}(M - \lambda_j(M)I)$, so that the linear systems (3.1) become ill conditioned. For a simple eigenvalue λ_j and random and properly scaled vectors $\mathbf{u}_i$ and $\mathbf{v}_i$, additive preprocessing $M - \lambda^{(i)}I \rightarrow K_i = M - \lambda^{(i)}I + \mathbf{u}_i \mathbf{v}_i^H$ is expected to yield a well conditioned matrix K_i [25], [28], and we can replace stage (3.1) by computing either the vector $\mathbf{y}_i = K_i^{-1}(1 + g_i^{-1} \mathbf{u}_i \mathbf{v}_i^H K_i^{-1})\mathbf{w}_i$ for $g_i = 1 - \mathbf{v}_i^H K_i^{-1} \mathbf{u}_i$ or $\mathbf{y}_i = K_i^{-1} \mathbf{u}_i$. We call the two resulting algorithms the *SMW* and *AP* iterations, respectively, each having the RQ and SQ versions.

For an eigenvalue $\lambda^{(i)} = \lambda_j(M)$ both *SMW* and *AP* iterations compute an associated eigenvector $\mathbf{y}_i$, due to the SMW formula and Corollary 2.1, respectively. Both SMW and AP iterations inherit local quadratic and rapid global convergence of the RQ and SQ iterations [28].

REMARK 3.1. (a) *The algorithms in this section similarly approximate an eigenvalue $\lambda_j(M)$ having geometric multiplicity $\nu > 1$ provided $\text{rank}(UV^H) = \nu$ for a pair of properly scaled random $n \times \nu$ matrices U and V , which we can vary from step to step. The multiplicity ν can be computed via the linear or binary search as the minimum integer for which the matrix $K_i = M - \lambda^{(i)}I + UV^H$ is nonsingular and well conditioned. To test if it is, we can just verify a fixed tolerance bound $\|K_i Y_i - U\| \leq t\|U\|$ on the relative residual norm of the matrix $Y_i = K_i^{-1}U$, which we compute numerically with the standard IEEE double precision. Wherever the matrix K_i passes this test, we obtain the matrix Y_i as by-product to be used in our eigen-solver. (b) Moreover all our eigen-solvers (both with and without additive preprocessing) can be readily extended to approximating a cluster or any fixed set of ν simple eigenvalues for $1 \leq \nu \ll n$. One just needs to replace the matrices $M - \lambda^{(i)}I$ with $\prod_{j=1}^{\nu} (M - \lambda_j^{(i)}I)$ where $\lambda_1^{(i)}, \dots, \lambda_{\nu}^{(i)}$ are the current approximations to the ν fixed eigenvalues and to update such approximations based on the Rayleigh–Ritz procedure in [4], [30] (instead of RQs or SQs).*

Newton's iteration relying on our next results is a reliable alternative near the eigenvalues. We refer to the respective

quadratically converging algorithm as **Algorithm 3.1**. See some details in [27].

THEOREM 3.1. *Let U_i, V_i, X, Y, X_i, Y_i be $n \times \nu$ matrices, $0 < \nu < n$. Let a triple $\{\lambda^{(i)}, X_i, Y_i\}$ approximate an eigentriple $\{\lambda, X, Y\}$ of an $n \times n$ matrix M where the eigenvalue λ has algebraic and geometric multiplicity ν . Write $M(\lambda) = M - \lambda I$, $M_i = M - \lambda^{(i)}I$, $K = M + UV^H$, $K(\lambda) = K - \lambda I$, $K_i = K - \lambda^{(i)}I$, $\delta^{(i)} = \lambda - \lambda^{(i)}$, and $\Delta_i = Y - Y_i$ and suppose $K_i^H X_i = V$, $K_i Y_i = U$, and the matrices $K(\lambda)$ and K_i are nonsingular. Then $X^H K(\lambda) = V^H$, $K(\lambda)Y = U$, $\Delta_i = \delta^{(i)} K_i^{-1}(I - (\delta^{(i)}) K_i^{-1})^{-1} Y_i = \delta^{(i)} K_i^{-1} Y_i + O(|\delta^{(i)}|^2)$, and $M_i Y_i = \delta^{(i)} U T_i + O(|\delta^{(i)}|^2)$ where $T_i = V^H K_i^{-2} U$.*

COROLLARY 3.1. *Under the assumptions of Theorem 3.1, let the matrix U have full column rank. Then $\delta^{(i)} T_i = G_i + O(|\delta^{(i)}|^2)$ where $G_i = I_{\nu} - V^H K_i^{-1} U$ and $T_i = V^H K_i^{-2} U + O(|\delta^{(i)}|^2)$.*

4. POLYNOMIAL ROOT-FINDING USING COMPANION MATRICES

4.1 Companion matrix and its eigenspaces

To approximate the roots of a polynomial $p(x) = \prod_{j=1}^n (x - \lambda_j)$, we can apply the algorithms in the previous section to the associated companion matrix $F_{\mathbf{p}}$ in (2.2) and exploit its structure to accelerate the computations dramatically.

FACT 4.1. *Any eigenvalue $\lambda_j = \lambda_j(F_{\mathbf{p}})$ has a left eigenvector $\mathbf{y}_j^T = (\lambda_j^{i-1})_{i=1}^n$.*

COROLLARY 4.1. *If the companion matrix $F_{\mathbf{p}}$ has n distinct simple eigenvalues $\lambda_j = \lambda_j(F_{\mathbf{p}})$ for $j = 1, 2, \dots, n$, then $Y F_{\mathbf{p}} Y^{-1} = \text{diag}(\lambda_j)_{j=1}^n$ where $Y = (\mathbf{y}_j^T)_{j=1}^n = (\lambda_j^{i-1})_{j,i=1}^n$.*

FACT 4.2. *Suppose $r(x) = \frac{u(x)}{w(x)}$, $u(x)$ and $w(x)$ are two polynomials, and the matrix $w(F_{\mathbf{p}})$ is nonsingular. Then the matrices $F_{\mathbf{p}}$ and $r(F_{\mathbf{p}})$ share the eigenvectors associated with their eigenvalues λ_j and $r(\lambda_j)$ for $j = 1, 2, \dots, n$, respectively.*

The polynomials $p_{\text{rev}}(x) = \frac{1}{p_0} x^n p(1/x)$, $p_0 \neq 0$, and $p(x - \mu) = q(x) = \sum_{i=0}^n q_i x^{n-i}$ have the roots $1/\lambda_j$ and $\lambda_j + \mu$ for $j = 1, 2, \dots, n$ and the coefficient vectors $\mathbf{p}_{\text{rev}}$ and $\mathbf{q}$, respectively. The matrices $F_{\mathbf{p}_{\text{rev}}}$ and $F_{\mathbf{q}}$ share their eigenvalues but not eigenspaces with the matrices $F_{\mathbf{p}}^{-1}$ and $F_{\mathbf{p}} - \mu I$, respectively. For $p_0 \neq 0$ we have $F_{\mathbf{p}}^{-1} = Z^T - (\frac{p_1}{p_0})_{i=1}^n \mathbf{e}_1^T = J F_{\mathbf{p}_{\text{rev}}} J$.

Computation of the coefficients of the polynomial $p(x - \mu)$ takes $O(n \log n)$ ops (see, e.g., [20]). Hereafter v_M (resp. v_M^{-1}) denotes the minimum number of ops required for multiplication of a matrix M by a vector (resp. solving the linear system $M\mathbf{y} = \mathbf{w}$ for a nonsingular matrix M). Clearly, $v_{F_{\mathbf{p}}} < 2n$, $v_{F_{\mathbf{p}}^{-1}} < 2n$, and $v_{F_{\mathbf{p}} - \mu I} < 4n$. Furthermore $v_{(F_{\mathbf{p}} - \mu I)^{-1}} < 7n - 7$ (apply Gaussian elimination), $v_{(F_{\mathbf{p}} - \mu I + \mathbf{p}\mathbf{e}_n^T)^{-1}} < 2n$ for $\mu \neq 0$ since $F_{\mathbf{p}} - \mu I + \mathbf{p}\mathbf{e}_n^T = Z - \mu I$ (cf. Lemma 2.1), and similarly $v_{(F_{\mathbf{p}} + (\mathbf{p} + \mu \mathbf{e}_n)\mathbf{e}_n^T - \mu I)^{-1}} < 2n$.

4.2 Accelerated SQ iteration

Due to these bounds every RQ iteration step needs at most $12n$ ops, not counting ops for testing the stopping criterion and computing the scalar c_i . We can save ops by using the

SQ iteration and by adding the preprocessor $\mathbf{p}\mathbf{e}_n^T$. With this preprocessor the RQ and SQ iterations produce crude but reasonable approximations in as few steps as with no preprocessing, but then fail to refine them. At this point we should shift to the RQ or SQ iteration with no preprocessing, to SMW iteration, or if the current approximation is close enough to an eigenvalue, then to Algorithm 3.1 with the same preprocessor $\mathbf{p}\mathbf{e}_n^T$ and with the steps simplified respectively. We refer to the TR and journal versions of this paper and to paper [27] on the details, test results (showing the power of our algorithms), and some recipes for the initialization, deflation, etc.

4.3 Repeated squaring

Write $F^{(0)} = F_{\mathbf{p}}$ and recursively compute the matrices $F^{(i+1)} = (F^{(i)})^2$ for $i = 0, 1, \dots$. The impact of k repeated squarings is the same as of 2^k steps of the classical Power method [12, Section 7.3.1].

Squaring is closed in the class of rational matrix functions $r(F_{\mathbf{p}})$ (cf. Fact 4.2), whose pairwise multiplication can be reduced essentially to a small number of FFTs and performed in $O(n \log n)$ ops. Moreover this computation is numerically stable if the input matrices are represented in the so called Horner's basis (see [10], [22, Section 6]). A matrix $r(F_{\mathbf{p}})$ has Toeplitz-like structure, has displacement rank at most two, and so can be inverted in $O(n \log^2 n)$ ops if it is nonsingular (cf., e.g., [20, Chapter 5]). The initial squaring steps are even less costly.

THEOREM 4.1. *The i th squaring of the matrices $F_{\mathbf{p}}, F_{\mathbf{p}}^T, F_{\mathbf{p}}^{-1}$, and $F_{\mathbf{p}}^{-T}$ (with no shift) uses at most $2^{2i+1}n$ multiplications and $4^i(2n-1)$ additions for $i \leq \log_2 n$.*

THEOREM 4.2. $\lim_{i \rightarrow \infty} \mathbf{e}_j^T F^{(i)} \in \mathbb{S}_1$ for the left eigenspace $\mathbb{S}_1$ of a companion matrix $F_{\mathbf{p}}$ generated by its eigenvector $(\lambda_1^h)_{h=0}^{n-1}$ provided $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_{n-1}| \geq |\lambda_n|$ and $\lambda_j = \lambda_j(F_{\mathbf{p}})$ for $j = 1, \dots, n$ denote the n eigenvalues of the matrix $F_{\mathbf{p}}$.

COROLLARY 4.2. *Under the assumptions of Theorem 4.2, $\lim_{i \rightarrow \infty} \frac{\mathbf{e}_j^T F^{(i)} \mathbf{e}_{k+1}}{\mathbf{e}_j^T F^{(i)} \mathbf{e}_k} = \lambda_1$ for every pair of integers j and k such that for $1 \leq j \leq n$, $1 \leq k < n$, and $\mathbf{e}_j^T F^{(i)} \mathbf{e}_k \neq 0$. If $|\lambda_{n-1}| > |\lambda_n|$, then $\lim_{i \rightarrow \infty} \mathbf{w}^{(i)} = (F^{(i)})^{-T} \mathbf{u} \in \mathbb{S}_1$ and $\lim_{i \rightarrow \infty} (F^{(i)T} + \mathbf{u}\mathbf{v}^H)^{-1} \mathbf{u} \in \mathbb{S}_1$ (cf. Theorem 2.1), so that $\lim_{i \rightarrow \infty} \frac{\mathbf{w}^{(i)T} \mathbf{e}_{k+1}}{\mathbf{w}^{(i)T} \mathbf{e}_k} = \lambda_n$.*

Convergence to the eigenvalue λ_1 (resp. λ_n) and the associated eigenspace is fast unless $|\frac{\lambda_1}{\lambda_2}| \approx 1$ (resp. $|\frac{\lambda_{n-1}}{\lambda_n}| \approx 1$). In our tests with random input polynomials $p(x)$ convergence to the absolutely smallest eigenvalue λ_n was faster than to the absolutely largest eigenvalue λ_1 .

4.4 Real eigen-solving

We first map the real roots of an input polynomial $p(x)$ into the unit circle $C_1 = \{x : |x| = 1\}$ by shifting from the matrix $F_{\mathbf{p}}$ to the matrix $F^{(0)} = I + 2\sqrt{-1}(F_{\mathbf{p}} - \sqrt{-1}I)^{-1}$. Next we propose an approach to approximating the eigenvalues of the matrix $F^{(0)}$ lying on the circle C_1 .

Write $F_i = \frac{1}{2}(F^{(i)} + (F^{(i)})^{-1}) = \frac{1}{2}(I + F^{(i+1)})(F^{(i)})^{-1}$, $\lambda_j^{(i)} = \lambda_j(F^{(i)})$, and $\lambda_{j,i} = \lambda_j(F_i)$ for $j = 1, \dots, n$; $i = 1, 2, \dots$, and observe the following simple facts.

FACT 4.3. $F_i = \frac{1}{2}((F + \sqrt{-1}I)^{2^i} + (F - \sqrt{-1}I)^{2^i})(F^2 + I)^{-2^{i-1}}$ for $F = F_{\mathbf{p}}$ and $i = 1, 2, \dots$. In particular, $F_1 = (F^2 - I)(F^2 + I)^{-1}$, $F_2 = (F^4 - 6F^2 + I)(F^2 + I)^{-2}$.

FACT 4.4. *The eigenvalues of the matrices F_i are equal to $\frac{1}{2}(\lambda_j^{(i)} + (\lambda_j^{(i)})^{-1}) = \frac{1}{2}((\lambda_j^{(0)})^{2^i} + (\lambda_j^{(0)})^{-2^i})$ for $j = 1, \dots, n$ and all i .*

FACT 4.5. *For every integer i the matrix F_i shares its eigenspaces with the matrix $F_{\mathbf{p}}$.*

FACT 4.6. *The map $F_{\mathbf{p}} \rightarrow F_i$ moves the real eigenvalues into the unit disc $D_1 = \{x : |x| \leq 1\}$.*

FACT 4.7. *Either $|\lambda_j^{(i)}| = 1$ for all i and j or the eigenspace $\mathbb{S}_{\mathbb{R}}$ associated with all the real eigenvalues of the matrix $F_{\mathbf{p}}$ is a dominated eigenspace of the matrices F_i for all sufficiently large integers i .*

Facts 4.3–4.7 together with Theorem 2.1 imply the following corollary.

COROLLARY 4.3. *Unless $|\lambda_j^{(i)}| = 1$ for all i and j we have $\lim_{i \rightarrow \infty} \mathcal{R}(K_i^{-1}U_i) \in \mathbb{S}_{\mathbb{R}}$ provided $r = \dim \mathbb{S}_{\mathbb{R}}$, $K_i = F_i$ or $K_i = F_i + U_i V_i^H$, K_i are nonsingular and well conditioned matrices, and U_i and V_i are scaled $n \times r$ matrices such that $\min_{\mathbf{s} \in \mathbb{S}_{\mathbb{R}}} \frac{\|U_i^T \mathbf{s}\|}{\|U_i\|} \geq b$ for a fixed positive constant b and $i \rightarrow \infty$.*

To obtain the integer r we can apply the Sturm sequence to the polynomial $p(x)$ or the binary search based on Corollary 2.1 to the matrices $M = F_i$ for larger integers i . The algorithms in [29] (cf. also [21]) estimate all root radii $|\lambda_j(F)|$ at a low cost.

Now write $\theta_i = \min_{j: |\lambda_j(F_i)| > 1} \max\{|\lambda_j^{(i)}|, 1/|\lambda_j^{(i)}|\} = \theta_0^{2^i}$ and let not all roots of the polynomial $p(x)$ be real, so that $\theta_0 > 1$. Then for larger integers i and a matrix $K_i^{-1}U_i$ in Corollary 4.3, the space $\mathcal{R}(K_i^{-1}U_i)$ approximates the eigenspace $\mathbb{S}_{\mathbb{R}}$ much closer than the eigenspaces associated with the remaining eigenvalues.

Therefore we can first refine this approximation by applying the inverse Rayleigh–Ritz iteration to the matrix F_i and then deflate the matrix $F_{\mathbf{p}}$, decoupling its $r \times r$ block L , whose r eigenvalues are precisely the r real eigenvalues of the matrix $F_{\mathbf{p}}$, associated with the eigenspace $\mathbb{S}_{\mathbb{R}}$. (See [4], [30] on the inverse Rayleigh–Ritz iteration and deflation.)

Besides decreasing the size of the original problem from n to r , we get rid of nonreal roots and can apply either the algorithm in [7] or Laguerre's (or quasi-Laguerre's) algorithm [18], [32]. Both of them are proved to be highly effective where all output values are real.

In our tests, however, we observed rapid convergence of the RQ or SQ iterations even where we initialized them at the origin and applied to one of the matrices F_0, F_1 and F_2 . More precisely in this case we only applied the iteration until we satisfied our stopping criterion with the tolerance 10^{-2} , and then we used the computed approximate eigenvector (shared by the matrices F_i and $F_{\mathbf{p}}$) to initialize the application of the same iteration to the matrix $F_{\mathbf{p}}$, in which case we set tolerance to 10^{-6} . Hereafter we refer to this two-stage algorithm as **Algorithm 4.1**. Whenever the process converged to a nonreal eigenvalue (this occurred in less than 20% of the runs in our tests), we deflated it together with its complex conjugate eigenvalue and reapplied the same algorithm to the deflated matrix of dimension $n-2$.

In Appendix A we present a matrix-free variant of Algorithm 4.1 for real root-finding.

5. POLYNOMIAL ROOT-FINDING VIA DPR1 EIGEN-SOLVING

For a polynomial $p(x)$ of a degree n we define its generalized companion matrices

$$C = C_{\mathbf{s}, \mathbf{u}, \mathbf{v}} = D_{\mathbf{s}} - \mathbf{u}\mathbf{v}^H \quad (5.1)$$

for $\mathbf{s} = (s_i)_{i=1}^n$, $\mathbf{u} = (u_i)_{i=1}^n$, $\mathbf{v} = (v_i)_{i=1}^n$,

$$D_{\mathbf{s}} = \text{diag}(s_i)_{i=1}^n, \prod_i s_i \neq 0, \quad (5.2)$$

$$d_i = u_i v_i = \frac{p(s_i)}{q_i(s_i)} \neq 0, \quad q_i(x) = \prod_{j \neq i} (x - s_j), \quad i = 1, \dots, n, \quad (5.3)$$

$$q_i(s_i) = q'(s_i), \quad i = 1, \dots, n, \quad q(x) = \prod_{j=1}^n (x - s_j). \quad (5.4)$$

C and $C - \mu I$ for scalars μ are diagonal+rank-one (hereafter DPR1) matrices. Unlike the companion matrices, they are defined by the values of the associated polynomial on a fixed set of points rather than by the coefficients.

THEOREM 5.1. (See, e.g., [11], [6, Theorem 4.4], [7].) *The eigenvalues of the matrix C in (5.1) are precisely the roots of the polynomial $p(x)$ as well as the secular equation*

$$\sum_{i=1}^n \frac{u_i v_i}{s_i - \lambda} = 1. \quad (5.5)$$

THEOREM 5.2. *Suppose we are given a scalar μ and $3n$ scalars u_i , v_i , and s_i , $i = 1, \dots, n$, which define a DPR1 generalized companion matrix C in equation (5.1). Write $s = 1 - \sum_{i=1}^n \frac{u_i v_i}{s_i}$ and let $s \neq 0$. (For $s = 0$ equation (5.5) has the root $\lambda = 0$.) Then we can compute $3n$ parameters $u_i^{(\text{new})}$, $v_i^{(\text{new})}$, and $s_i^{(\text{new})}$, $i = 1, \dots, n$, that define a DPR1 generalized companion matrices (a) $C - \mu I$ in n ops, (b) C^{-1} in $6n$ ops (where the matrix C is nonsingular), and (c) C_{rev} associated with the polynomial $p_{\text{rev}}(x)$ in $4n + 1$ ops.*

PROOF. Part (a) is obvious. Part (b) follows from the SMW formula. (c) To define the matrix C_{rev} , we seek $3n$ parameters $u_i^{(\text{new})}$, $v_i^{(\text{new})}$, and $s_i^{(\text{new})}$, $i = 1, \dots, n$, such that

$$\sum_{i=1}^n \frac{d_i^{(\text{new})}}{s_i^{(\text{new})} - (1/\lambda)} = 1 \quad (5.6)$$

for $d_i^{(\text{new})} = u_i^{(\text{new})} v_i^{(\text{new})}$ and for all values λ satisfying equation (5.5). First rewrite equation (5.6) as $\sum_{i=1}^n \frac{d_i^{(\text{new})} \lambda}{s_i^{(\text{new})} \lambda - 1} =$

1. Then substitute the expressions $\frac{d_i^{(\text{new})} \lambda}{s_i^{(\text{new})} \lambda - 1} = \frac{d_i^{(\text{new})}}{s_i^{(\text{new})}} (1 + \frac{1}{s_i^{(\text{new})} \lambda - 1})$ for $i = 1, \dots, n$ and obtain that equation (5.6)

is equivalent to the equation $\sum_{i=1}^n \frac{d_i^{(\text{new})}}{s_i^{(\text{new})}} \frac{1}{s_i^{(\text{new})} \lambda - 1} = s^{(\text{new})}$

for $s^{(\text{new})} = 1 - \sum_{i=1}^n \frac{d_i^{(\text{new})}}{s_i^{(\text{new})}}$. Now write $s_i^{(\text{new})} = 1/s_i$,

$d_i^{(\text{new})} = -s^{(\text{new})} d_i / s_i^2$ for $i = 1, \dots, n$ and observe that under this assignment $s^{(\text{new})} = 1/s$ and equations (5.5) and (5.6) are equivalent to one another. It remains to compute $s_i^{(\text{new})} = 1/s_i$ (in n ops), $w_i = d_i / s_i$ (in n ops), $u_i^{(\text{new})} = w_i / s_i$ (in n ops) for $i = 1, \dots, n$, $-s = \sum_{i=1}^n w_i - 1$ (in n ops), $v_i^{(\text{new})} = -1/s$ for $i = 1, \dots, n$. $\square$

Next we employ these results and the DPR1 matrix structure to extend the eigen-solving approaches in the previous section to the matrices C in (5.1).

THEOREM 5.3. *DPR1 matrix C in (5.1) has the eigendecomposition $C = W^{-1} D_{\Lambda} W$ provided $D_{\Lambda} = \text{diag}(\lambda_j)_{j=1}^n$, $W = (\frac{u_i}{s_i - \lambda_j})_{i,j=1}^n$, $W^{-1} = (\frac{v_j}{s_i - \lambda_j})_{i,j=1}^n$, and the scalars in the n -tuple $\{s_1, \dots, s_n\}$ (as well as $\{\lambda_1, \dots, \lambda_n\}$) are distinct.*

PROOF. The j th row of the matrix W^{-1} and the j th column of the matrix W are the left and right null vectors of the matrix $C - \lambda_j I$, respectively. Apply Theorem 2.1. $\square$

The following result relies on the SMW formula (2.1) for $r = 1$ (see [6, Theorem 5.1]).

THEOREM 5.4. *For the DPR1 matrix C in equation (5.1) with the vector $\mathbf{u} = (\pm 1, \pm 1, \dots, \pm 1)^T$, a scalar μ , and a vector $\mathbf{w}$, we can compute the vector $(C - \mu I)^{-1} \mathbf{w}$ by performing $9n$ ops provided the matrix $C - \mu I$ is nonsingular.*

With additive preprocessing we decrease the cost bound.

THEOREM 5.5. *For a scalar μ , four vectors $\mathbf{s}$, $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ of dimension n , and the DPR1 matrix C in equation (5.1), let the matrix $K = C + \mathbf{u}\mathbf{v}^H - \mu I = D_{\mathbf{s}} - \mu I$ be nonsingular. Then we can compute the vector $K^{-1} \mathbf{w}$ in $2n$ ops.*

As a stopping criterion we can just check whether secular equation (5.5) is satisfied for a fixed scalar λ within a fixed tolerance bound. This takes $2n$ ops assuming that the products $d_i = u_i v_i$ have been given to us for all i . Moreover these ops can be reused when we update any approximate eigenvalue $\lambda^{(k)}$ according to the formula

$$\lambda^{(k+1)} = \lambda^{(k)} + \frac{1 - \beta_k^{(1)}}{\beta_k^{(2)}}, \quad \beta_k^{(h)} = \sum_{j=1}^n \frac{d_j}{(s_j - \lambda^{(k)})^h} \quad (5.7)$$

provided $h = 1, 2$ and $s_j \neq \lambda^{(k)}$ for all pairs $\{j, k\}$.

Apply Theorem 3.1 to a DPR1 matrix C and obtain

COROLLARY 5.1. *We have $|\lambda^{(k+1)} - \lambda| = O(|\lambda^{(k)} - \lambda|^2)$ as $|\lambda^{(k)} - \lambda| \rightarrow 0$ for $\lambda^{(k+1)}$ in equation (5.7) and an eigenvalue λ of the matrix C in (5.1).*

We can update an approximate eigenvalue $\lambda^{(k)}$ via equation (5.7) in $5n + 1$ ops, including the $2n$ ops reused from verifying secular equation (5.5).

In the case of DPR1 matrices, we can begin with crude approximations $s_1, \dots, s_n$ to the eigenvalues and then recursively update them (together with the DPR1 matrix based on equations (5.1)–(5.4)) as we improve the approximations. To update a single s_i we use $9n - 8$ ops. See [16] and [11] on the power of such an updating technique.

Initially DPR1 squarings are not costly (similarly to the squarings of companion matrix in Theorem 4.1), but gradually the DPR1 structure deteriorates in squaring. Our next theorem (for $r = 2$ and DPR2 matrix $(D - \mathbf{u}\mathbf{v}^H)^2$) alleviates the problem. It enables fast implicit squaring although does not preserve the eigenvectors.

We state this theorem in a more general form than we need in this paper. First we define $n \times n$ DPR r matrices as diagonal + rank- r matrices of the form $C = D - UV^H$

where D is an $n \times n$ diagonal matrix and U and V are $n \times r$ matrices. If both matrices C and D are nonsingular we can apply the SMW formula and deduce that C^{-1} is a DPRr matrix as well. Furthermore we immediately observe that $C_1 C_2 = D_1 D_2 - U_1 \tilde{V}_1 - \tilde{U}_2 V_2$ where $C_j = D_j - U_j V_j^H$ are DPRr $_j$ matrices for $j = 1, 2$, $\tilde{V}_1 = V_1^H (D_2 - U_2 V_2^H)$ and $\tilde{U}_2 = D_1 U_2$, so that $C_1 C_2$ is a DPR($r_1 + r_2$) matrix.

THEOREM 5.6. (a) Let $s_1, \dots, s_n; \mu_1, \dots, \mu_n$ be $2n$ distinct scalars. Define the diagonal matrix $D = \text{diag}(s_i)_{i=1}^n$, a pair of $n \times r$ matrices U and V , and the DPRr matrix $C = D - UV^H$. Then it is sufficient to use $O((r^3 + \log^2 n)n)$ ops to compute the values $\det(C - \mu_h I)$ for $h = 1, \dots, n$. (b) Furthermore $O((r^3 + \log n)n)$ ops are sufficient if $s_i = \omega_k^{i-1}$ and $\mu_h = b\omega_l^{h-1}$ for $h, i = 1, \dots, n$ and two nonzero scalars a and b , where ω_q denotes a primitive q -th root of unity, $k \geq n$, $l \geq n$, and $k + l = O(n)$.

PROOF. We have $\det(C - \mu_h I) = (\det(D - \mu_h I)) \det(I_r - V^H (D - \mu_h I)^{-1} U)$ (cf. e.g., [23]). Within the claimed cost bounds we compute at first the coefficients and then the values $\det(D - \mu_h I) = \prod_{j=1}^n (s_j - \mu_h)$ of the polynomial $\prod_{j=1}^n (s_j - x)$ at the n points $x = \mu_h$, $h = 1, \dots, n$ (cf. [20, Section 3.1]). It remains to compute the $r \times r$ matrices $I_r - G_h = V^H (D - \mu_h I)^{-1} U = (\sum_{j=1}^n \frac{v_{ij} u_{jk}}{s_j - \mu_h})_{i,k=0}^{r-1}$, $h = 1, \dots, n$ (in $O(nr^2)$ ops), then the $r \times r$ matrices G_h for $h = 1, \dots, n$ (in nr ops), and finally the n values of their determinants (in $(4r^3 - 15r^2 + 23r - 6)n/6$ ops by means of Gaussian elimination). $\square$

The theorem can be extended to bidiagonal and tridiagonal matrices. It bounds the cost of computing the values of the characteristic polynomial $\det(\mu_h I - C)$ of the matrix C at n points $\mu_1, \dots, \mu_n$. Within the same cost bound we can compute the coefficients of the polynomial $q(x) = \prod_{h=1}^n (x - \mu_h)$ and the values $q'(\mu_h)$ for all h , thus defining a DPR1 matrix $D + \mathbf{u}\mathbf{v}^H$. It shares its eigenvalues with the matrix C . For $r = 2$ we compute (in $O(n \log^g n)$ ops for $g \leq 2$) a DPR1 matrix whose eigenvalues are the squares of the eigenvalues of a given DPR1 matrix, and so we can extend the repeated squaring from Section 4.3. Numerically the approach has deficiency of employing the characteristic polynomial rather than the eigenvectors and eigenspaces.

In squaring we can choose the values s_i and μ_h to our advantage, e.g., $s_i = \omega_{3^k}^{i-1}$ for $i = 1, 2, \dots, n$, and an integer k such that $n \leq 3^k < 3n$. Then in all squarings $s_i^{2^g} = \omega_{3^k}^{(i-1)j}$ for $i = 1, 2, \dots, n$, $j = 2^g \bmod 3^k$, $g = 1, 2, \dots$, and we can apply part (b) of Theorem 5.6. We can modify the squaring stages in our algorithm to compute cubic powers rather than squares, and then we can choose $s_i^{2^g} = \omega_{2^k}^{(i-1)j}$ for $i = 1, 2, \dots, n$, $j = 2^g \bmod 2^k$, $g = 1, 2, \dots$, and employ the FFT subroutines.

6. NUMERICAL TESTS

We performed a series of numerical experiments in the Graduate Center of the City University of New York. We used a Dell server with a dual core 1.86 GHz Xeon processor and 2G memory running Windows Server 2003 R2. The test Fortran code was compiled with the GNU gfortran compiler within the Cygwin environment.

We generated random numbers with the random_number intrinsic Fortran function under the uniform probability distribution over the range $\{x : 0 \leq x < 1\}$. To shift to the

range $\{y : b \leq y \leq a + b\}$ for real a and b , we applied the linear transform $x \rightarrow y = ax + b$.

Tables 1–6 display the results of our tests of the real eigen-solving Algorithm 4.1 assuming $n \times n$ companion input matrix for $n = 64, 128, 256$ and using 0, 1, and 2 squarings. Tables 2, 4, and 6 represent Stage 2, at which approximations given with errors below 10^{-2} are refined to decrease the errors below 10^{-6} . We run 100 tests for each input size and display the average (mean), maximum, and minimum numbers of iteration loops until convergence as well as the standard deviations. The columns marked %rr display the percent of real roots among the roots computed by Algorithm 4.1.

The TR and journal versions display further test results, in particular for the AP iteration where $M = F_p$ are companion matrices, $\mathbf{u}_i = \mathbf{p}$ and $\mathbf{v}_i = \mathbf{e}_n$ as well as where $M = C$ are DPR1 matrices, $\mathbf{u}_i = \mathbf{u}$, $\mathbf{v}_i = \mathbf{v}$ for all integers i . The tests showed rapid initial decrease of the residual norm, but we needed shifting to Algorithm 4.1 (and sometimes initially to a few steps of Algorithms 3.1 or 3.2) to decrease it further.

Table 1: Numbers of RQ and SQ iteration loops in Algorithm 4.1 (no squaring, Stage 1)

RQ/SQ	n	min	max	mean	std
RQ	64	2	17	3.86	2.19
RQ	128	2	15	4.12	2.27
RQ	256	2	15	3.72	1.76
SQ	64	2	18	4.22	2.76
SQ	128	2	45	5.18	5.68
SQ	256	2	33	4.57	4.02

Table 2: Numbers of RQ and SQ iteration loops in Algorithm 4.1 (no squaring, Stage 2)

RQ/SQ	n	min	max	mean	std	%rr
RQ	64	0	2	0.99	0.32	90
RQ	128	0	2	0.9	0.43	89
RQ	256	0	2	0.82	0.42	82
SQ	64	0	2	1	0.37	91
SQ	128	0	2	0.97	0.38	91
SQ	256	0	2	0.88	0.45	90

Table 3: Numbers of RQ and SQ iteration loops in Algorithm 4.1 (one squaring, Stage 1)

RQ/SQ	n	min	max	mean	std
RQ	64	2	9	3.9	1.18
RQ	128	2	6	3.75	0.86
RQ	256	2	13	3.7	1.4
SQ	64	2	10	4.23	1.5
SQ	128	2	9	4.19	1.29
SQ	256	2	8	4.25	1.35

Table 4: Numbers of RQ and SQ iteration loops in Algorithm 4.1 (one squaring, Stage 2)

RQ/SQ	n	min	max	mean	std	%rr
RQ	64	0	17	2.44	2.27	89
RQ	128	0	6	2.39	1.28	89
RQ	256	0	14	2.51	1.79	83
SQ	64	0	14	2.3	2.01	86
SQ	128	0	10	2.46	1.81	90
SQ	256	0	12	2.9	2.32	83

Table 5: Numbers of RQ and SQ iteration loops in Algorithm 4.1 (two squarings, Stage 1)

RQ/SQ	n	min	max	mean	std
RQ	64	2	12	4.01	1.53
RQ	128	2	9	3.89	1.12
RQ	256	3	10	4.05	1.23
SQ	64	2	24	4.07	2.39
SQ	128	3	9	3.92	1.12
SQ	256	3	10	4.07	1.27

Table 6: Numbers of RQ and SQ iteration loops in Algorithm 4.1 (two squarings, Stage 2)

RQ/SQ	n	min	max	mean	std	%rr
RQ	64	0	4	1.51	0.72	91
RQ	128	0	6	1.71	1	91
RQ	256	0	8	2.06	1.43	88
SQ	64	0	4	1.62	0.79	90
SQ	128	0	16	2.33	2.49	90
SQ	256	0	12	2.27	1.88	83

APPENDIX

A. REAL ROOT-FINDER

ALGORITHM A.1. Real Root-finder

INPUT: a small positive tolerance τ , a positive integer n , and $n + 1$ real values $p_0, p_1, \dots, p_n, p_n \neq 0$, defining the polynomial $p(x) = \sum_{i=0}^n p_i x^i$.

OUTPUT: the integer r and the coefficients $v_0, v_1, \dots, v_r$ of the polynomial $v(x) = \sum_{i=0}^r v_i x^i$ such that $\deg b(x) < r$, $\|b(x)\| \leq \tau \|p(x)\|$ (assuming the polynomial norm $\|\sum_i s_i x^i\| = \sum_i |s_i|$), $v_r \neq 0$, $p(x) = v(x)q(x) + b(x)$, the polynomial $v(x)$ has r real roots counted with their multiplicities, and the polynomial $q(x)$ has no real roots.

INITIALIZATION: If $p(\sqrt{-1}) = p(-\sqrt{-1}) = 0$, set $n \leftarrow n - 2$ and $p(x) \leftarrow \frac{p(x)}{x^2 + 1}$. If $p(1) = 0$, set $n \leftarrow n - 1$ and $p(x) \leftarrow \frac{p(x)}{x - 1}$. Repeat until $p(\sqrt{-1})p(1) \neq 0$.

COMPUTATIONS:

1. Compute the polynomial $p_0(x) = \frac{(x - \sqrt{-1})^n}{p(1)} p(1 + \frac{2\sqrt{-1}}{x - \sqrt{-1}})$. ($p_0(x) = \prod_{j=1}^n (x - \lambda_j)$ for n unknown roots $\lambda_1, \dots, \lambda_n$.) (The real roots of $p(x)$ turn into the roots of $p_0(x)$ lying on the unit circle $C_1 = \{x : |x| = 1\}$.)
2. Fix a reasonably large integer k and apply k root-squaring steps of Dandelin (Lobachevsky, Gräffe), $p_{i+1}(x) = (-1)^n p_i(\sqrt{x}) p_i(\sqrt{-x})$, $i = 0, 1, \dots, k$, so that $p_i(x) = \prod_{j=1}^n (x - \lambda_j^{2^i})$ and the i -th iteration step squares the roots of the polynomial $p_{i-1}(x)$ for all i . Having performed k squaring steps, apply the algorithm in [29] (cf. [21]) to estimate the root radii $|\lambda_j|^{2^{k+1}}$ of the polynomial $p_{k+1}(x)$. Allow relative errors within a fixed tolerance δ (say within 0.01) and output the number r of the roots (counted with their multiplicity) that lie in the annulus $1 - \delta \leq |\lambda| \leq 1 + \delta$. If $r = n$, output $v(x) = p(x)$ and stop.
3. Otherwise as soon as the roots of the polynomial $p_k(x)$ lying on the unit circle C_1 become well separated from the other roots, apply [9, Algorithm 2.1] to the polynomial $p_k(x) = \prod_{j=1}^n (x - \lambda_j^{(k)})$, which should replace $p(x)$ in [9]. The algorithm outputs polynomial $\hat{p}(x) = \prod_{j=1}^n (x - \frac{1}{2}(\lambda_j^{(k)} + 1/\lambda_j^{(k)}))$ whose r absolutely smallest roots lie in the unit disc $D_1 = \{x : |x| \leq 1\}$.
4. Apply the algorithms in [29], [21] to compute and output an approximate factor $v(x)$ of the polynomial $\hat{p}(x)$ having these r roots.

Every squaring step as well as the root radii estimation takes $O(n \log n)$ ops, and so do Stage 1 (reduced to two variable shifts and the transition to the reverse polynomial between them (cf. [20, Problem 2.4.3])), Stage 3 (see [9], [19]), and Stage 4 provided that the roots in the unit disc D_1 are well separated from the other roots.

Computational precision tends to grow in root squaring, due to the widely uneven growth of the absolute values of the polynomial coefficients. We can safely perform a squaring step numerically by using the order of n^2 ops if the computation of a logarithm or an exponential is also counted as an op [17], but we would still need to compute with extended precision at Stage 3.

We can, however, reuse the remedy from the previous section, that is stop Stage 2 for a smaller integer k , say for $k \leq 2$, and instead of performing Stage 4, apply to the polynomial $\hat{p}(x)$ Müller's or Newton's iteration initiated at or near the origin. We can expect that it converges to a root of the polynomial $p_k(x)$ lying in the unit disc D_1 because such roots tend to be closest to the origin. Having approximated such a root y of $\hat{p}(x)$, we recover the respective root $x = z$ of the polynomial $p_0(x)$ satisfying $z^2 + 2zy + 1 = 0$, and then the root $\lambda = \sqrt{-1} \frac{1-y}{1+y}$ of the input polynomial $p(x)$. We would output this root if it is real. Otherwise, if $\lambda = r + s\sqrt{-1}$ for real r and $s \neq 0$, we would deflate the polynomial $p(x)$ by dividing it by $x^2 - 2rx + r^2 + s^2$ and would reapply the algorithm.

B. REFERENCES

- [1] A. Amiraslani, P. Lancaster, Rayleigh Quotient Algorithms for Nonsymmetric Matrix Pencils, *Numerical Algorithms*, **51**, 1, 5–22, 2009.
- [2] D. A. Bini, Numerical Computation of Polynomial Zeros by Aberth's Method, *Numerical Algorithms*, **13**, 179–200, 1996.
- [3] D. A. Bini, P. Boito, Y. Eidelman, L. Gemignani, I. Gohberg, A Fast Implicit QR Eigenvalue Algorithm for Companion Matrices, *Linear Algebra and Its Applics.*, **432**, 2006–2031, 2010.
- [4] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, 2000.
- [5] D. A. Bini, Y. Eidelman, L. Gemignani, I. Gohberg, Fast QR Eigenvalue Algorithms, *SIMAX*, **29**, 2, 566–585, 2007.
- [6] D. A. Bini, L. Gemignani, V. Y. Pan, Inverse Power and Durand/Kerner Iteration for Univariate Polynomial Root-finding, *Computers and Math. (with Applics.)*, **47**, 2/3, 447–459, 2004. (Also TR 2002 020, *Computer Science Department, Graduate Center, CUNY*, 2002.)
- [7] D. A. Bini, L. Gemignani, V. Y. Pan, Fast and Stable QR Eigenvalue Algorithms for Generalized Companion Matrices and Secular Equation, *Numerische Math.*, **3**, 373–408, 2005. (Also TR 1470, *Math. Department, University of Pisa*, Italy, July 2003.)
- [8] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.
- [9] D. Bini, V. Y. Pan, Graeffe's, Chebyshev, and Cardinal's Processes for Splitting a Polynomial into Factors, *J. Complexity*, **12**, 492–511, 1996.
- [10] J. P. Cardinal, On Two Iterative Methods for Approximating the Roots of a Polynomial, *Lectures in Applied Mathematics*, **32**, 165–188, AMS, Providence, RI, 1996.
- [11] S. Fortune, An Iterated Eigenvalue Algorithm for Approximating Roots of Univariate Polynomials, *J. of Symbolic Computation*, **33**, 5, 627–646, 2002. (Also in *ISSAC 2001*.)
- [12] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 3rd edition, The Johns Hopkins Univ. Press, Baltimore, MD, 1996.
- [13] N. J. Higham, *Accuracy and Stability in Numerical Analysis*, 2nd edition, SIAM, 2002.
- [14] M. Hemmer, E. P. Tsigaridas, Z. Zafeirakopoulos, I. Z. Emiris, M. I. Karavelas, B. Mourrain, Experimental Evaluation of Univariate Real Solvers, in *Proc. SNC 2009*, Kyoto, 2009 (H. Kai, H. Sekigawa eds.), pages 105–113, ACM Press, NY, 2009.
- [15] A. Melman, A Unifying Convergence Analysis of Second-Order Methods for Secular Equations, *Math. Comp.*, **66**, 333–344, 1997.
- [16] F. Malek, R. Vaillancourt, Polynomial Zero-finding Iterative Matrix Algorithms, *Computers and Math. with Applics.*, **29**, 1, 1–13, 1995.
- [17] G. Malajovich, J. P. Zubelli, On the Geometry of Graeffe Iteration, *J. of Complexity*, **17**, 3, 541–573, 2001.
- [18] B. Parlett, Laguerre's Method Applied to the Matrix Eigenvalue Problem, *Math. of Computation*, **18**, 464–485, 1964.
- [19] V. Y. Pan, New Fast Algorithms for Polynomial Interpolation and Evaluation on the Chebyshev Node Set, *Computers and Math. (with Applics.)*, **35**, 3, 125–129, 1998.
- [20] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/NY, 2001.
- [21] V. Y. Pan, Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *J. Symbolic Comp.*, **33**, 5, 701–733, 2002. Proc. version in *ISSAC 2001*.
- [22] V. Y. Pan, Amended DSeSC Power Method for Polynomial Root-finding, *Computers and Math. with Applics.*, **49**, 9–10, 1515–24, 2005.
- [23] V. Y. Pan, D. Grady, B. Murphy, G. Qian, R. E. Rosholt, Schur Aggregation for Linear Systems and Determinants, *Theoretical Computer Science*, **409**, 255–268, 2008.
- [24] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Wang, X. Yan, Root-finding with Eigen-solving, pages 185–210 in *Symbolic-Numeric Computation*, (Dongming Wang, Li-Hong Zhi, eds.), Birkhäuser, Basel/Boston, 2007.
- [25] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, *Linear Algebra and Its Applics.*, **432**, 1070–89, 2010. Proc. version in *CSR 2008*.
- [26] V. Y. Pan, G. Qian, Randomized Preprocessing of Homogeneous Linear Systems, *Linear Algebra and Its Applics.*, **432**, 3272–3318, 2010.
- [27] V. Y. Pan, G. Qian, A. Zheng, Z. Chen, Matrix Computations and Polynomial Root-finding with Preprocessing, *Linear Algebra and Its Applics.*, in print.
- [28] V. Y. Pan, X. Yan, Additive Preconditioning, Eigenspaces, and the Inverse Iteration, *Linear Algebra and Its Applics.*, **430**, 186–203, 2009. Proc. version in *SNC 2007*.
- [29] A. Schönhage, The Fundamental Theorem of Algebra in Terms of Computational Complexity, manuscript, 1982.
- [30] G. W. Stewart, *Matrix Algorithms, Volume II: Eigensystems*, SIAM, 1998.
- [31] H. Unger, Nichtlinear Behandlung von Eigenwertaufgaben, *ZAMM Z. Andew. Math. Mech.*, **30**, 281–282, 1950.
- [32] X. Zou, Analysis of the Quasi-Laguerre Method, *Numerische Math.*, **82**, 491–519, 1999.

Computing the Radius of Positive Semidefiniteness of a Multivariate Real Polynomial Via a Dual of Seidenberg's Method*

Sharon Hutton
Dept. of Mathematics,
NCSU
Raleigh, North Carolina
27695-8205, USA
sehutton@ncsu.edu

Erich L. Kaltofen
Dept. of Mathematics, NCSU
Raleigh, North Carolina
27695-8205, USA
kaltofen@math.ncsu.edu
http://www.kaltofen.us

Lihong Zhi
Key Laboratory of Mathematics
Mechanization, AMSS
Beijing 100190, China
lzhi@mmrc.iss.ac.cn
http://www.mmrc.iss.ac.cn/~lzhi

ABSTRACT

We give a stability criterion for real polynomial inequalities with floating point or inexact scalars by estimating from below or computing the radius of semidefiniteness. That radius is the maximum deformation of the polynomial coefficient vector measured in a weighted Euclidean vector norm within which the inequality remains true. A large radius means that the inequalities may be considered numerically valid.

The radius of positive (or negative) semidefiniteness is the distance to the nearest polynomial with a real root, which has been thoroughly studied before. We solve this problem by parameterized Lagrangian multipliers and Karush-Kuhn-Tucker conditions. Our algorithms can compute the radius for several simultaneous inequalities including possibly additional linear coefficient constraints. Our distance measure is the weighted Euclidean coefficient norm, but we also discuss several formulas for the weighted infinity and 1-norms.

The computation of the nearest polynomial with a real root can be interpreted as a dual of Seidenberg's method that decides if a real hypersurface contains a real point. Sums-of-squares rational lower bound certificates for the radius of semidefiniteness provide a new approach to solving Seidenberg's problem, especially when the coefficients are numeric. They also offer a surprising alternative sum-of-squares proof for those polynomials that themselves cannot be represented by a polynomial sum-of-squares but that have a positive distance to the

nearest indefinite polynomial.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms; G.1.6 [Numerical Analysis]: Global optimization

General Terms

algorithms, experimentation

Keywords

numeric polynomial inequality, nearest polynomial with a real root, sum-of-squares, approximate polynomial systems, semidefinite programming

1. INTRODUCTION

1.1 Motivation

Real polynomial or rational function global optimization is equivalent to establishing a polynomial inequality: the infimum $\mu \in \mathbb{R}$ of a polynomial $f \in \mathbb{R}[x_1, \dots, x_n]$ satisfies $f(\xi_1, \dots, \xi_n) - \mu \geq 0$ for all $x_i = \xi_i \in \mathbb{R}$. In other words, the polynomial $f - \mu$ is positive semidefinite. For univariate f ($n = 1$) Sturm sequences [9] yield an efficient algorithm for deciding semidefiniteness. The bivariate case $n = 2$ can be solved by Seidenberg's [26] algorithm (see also [9] and [11]), which is generalized to arbitrarily many variables via Lagrangian multipliers in [1, 25] or used in nonstandard decision methods [29]. Alternatively, one can use Artin's theorem of sum-of-squares and semidefinite programming (see, e.g., [12, 14]).

Here we consider the more difficult situation when the coefficients of f are not exactly known, which is the case when f is the result of an empirical measurement or a computation with floating point numbers. As a simple example consider Figure 1 below.

The middle polynomial $f(x) = \frac{1}{3}(x-1)^2$ has a double real root at $x = 1$. In fact, it is the nearest polynomial with a real root to the polynomial $x^2 + 1$ under the infinity norm [8]: $\|f(x) - (x^2 + 1)\|_\infty = \frac{2}{3}$,

*This material is based on work supported in part by the National Science Foundation under Grants CCF-0830347, CCF-0514585, DMS-0532140 and OISE-0913588 (Hutton and Kaltofen).

Lihong Zhi is supported by the Chinese National Natural Science Foundation under Grants 60821002/F02, 60911130369 and 10871194.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

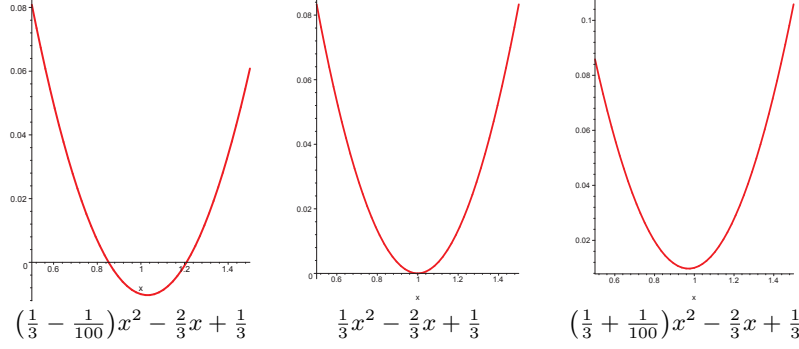


Figure 1: Root sensitivity

where for a polynomial g the norm $\|g\|_\infty$ is the maximum of the absolute values of the coefficients of g (the height of g). Small perturbations in the leading coefficient (one could also perturb the constant coefficient) make the polynomial f either indefinite (left polynomial in Figure 1, the polynomial changes sign) or positive definite (right polynomial). Therefore, the right polynomial, although positive definite, as an approximate polynomial is not numerically positive because a small change in its coefficients can make the polynomial indefinite. As in Kharitonov’s [16] interval polynomial stability criterion, we seek to compute by how much the coefficients in a polynomial can be deformed while still preserving nonnegativity. This distance is the coefficient vector norm distance to the nearest polynomial with a real root, which we shall call the *radius of positive semidefiniteness*. Note that there may not exist an affine optimizer—hence radius of positive semidefiniteness rather than distance to the nearest polynomial with a real root: the polynomial $(1 - \epsilon^2)x^2 + y^2 - 2xy + 4$ attains for any $\epsilon > 0$ negative values at $x = y > 2/\epsilon$. Thus the polynomial $x^2 + y^2 - 2xy + 4$ has a radius of positive semidefiniteness = 0 although its global minimum is 4.

1.2 Results and Used Approach

We follow the approach by Karmarkar and Lakshman [15] (see also [3]) which first fixes a real root $(\alpha_1, \dots, \alpha_n) \in \mathbb{R}^n$ and gives a rational function $\mathcal{N}(\alpha_1, \dots, \alpha_n)$ in the indeterminate α ’s for the minimal distance from the given f to the nearest polynomial $\tilde{f}$ with $\tilde{f}(\alpha_1, \dots, \alpha_n) = 0$. One then can compute the infimum of $\mathcal{N}(\alpha_1, \dots, \alpha_n)$ over all real α ’s. The case $n \geq 2$ is from [27].

We rederive the multivariate formula for $\mathcal{N}(\alpha_1, \dots, \alpha_n)$ in [27], for weighted ℓ^2 distance norms, by the method of Lagrangian multipliers. The weighted norms subsume the fixing of coefficients in [24] (see [4, Section 2.12.3.2.6] and Remark 5 below). Our approach also allows us to introduce linear constraints on the coefficients of $\tilde{f}$, as is done in [13] for the approximate GCD problem. Linear equality constraints generalize sparsity, which are equations of the form $c_i = 0$. Because the Jacobian of the Lagrange function remains linear in the control variables and multipliers, determinantal formulas parametric in the real root coordinates can be computed. Linear inequality constraints on the coefficients of $\tilde{f}$, for instance nonnegativity ($c_i \geq 0$) can now be imposed via Karush-Kuhn-Tucker (KKT) conditions (see, e.g., [5]) and the

arising systems solved via linear programming, at least for a fixed real root. Parametric root coordinates or non-linear constraints necessitate non-linear techniques on the Lagrange and KKT conditions and are therefore in general of much higher computational complexity. Our approach allows multiple simultaneous f ’s and complex coefficients without modification.

Our solution can be interpreted as a dual of Seidenberg’s test whether a real hypersurface f has a real point. Seidenberg’s algorithm (and Safey El Din’s generalization) computes to a given real point in $\mathbb{R}^n$ the nearest real point on f in terms of Euclidean distance. If f has no real solution the tangent equations have no real solutions. Our algorithm computes the nearest surface (in terms of coefficient norm) that has a real point. If f has a real point, the nearest surface is f itself. However, if a lower bound on the radius of semidefiniteness for any weight vector is > 0 , f has no real point, even when the coefficients of f are approximate. The latter can be certified by a sum-of-squares of rational functions, which leads to an entirely new verification that f is definite, i.e., has no real point, with possibly a very short certificate.

Polynomials with a radius of positive semidefiniteness > 0 are quite special. Our Example 6 below demonstrates that a positive polynomial that is not a sum-of-squares of polynomials can have a lower bound certificate for the radius of positive semidefiniteness that is in fact a sum-of-squares of polynomials, which implies positive semidefiniteness of the polynomial itself. For such polynomials, sum-of-squares denominators in Artin-style certificates may never become necessary (see our conjecture at the end of Section 5).

1.3 Related Previous Results

Our method is conceptually that of hybrid symbolic-numeric computation such as computing approximate polynomial greatest common divisors and factorization.

Hitz and Kaltofen [7] derive Lakshman’s and Karmarkar’s formula for univariate f by a least square fit for the cofactor $f(x)/(x - \alpha)$ and introduce linear equality constraints on the deformed coefficients. Zhi, Wu, Noda, Kai, Rezvan and Corless [31, 30, 23] generalize the formula to roots with given multiplicities. In [8] ℓ^∞ -norm distances are introduced and Markus Hitz in the Summer of 1999 considered dual ℓ^p -norms. Stetter [27] then generalized Lakshman and Karmarkar’s formula to

an arbitrary number of variables and dual ℓ^p norm distances via Hölders inequality.

In [24, 21] Stetter's multivariate (complex) formula is applied to the important problem of computing the nearest consistent polynomial system, with zeros of a minimum given multiplicity, and a different proof via generalized Lagrangian interpolation is given. We observe that the ℓ^∞ -norm formulas apply to the problem of consistent systems as well (see Theorem 7 below). In our setting, we determine the smallest deformation where all inequalities are simultaneously violated.

A related result [8] computes the nearest matrix in Frobenius norm that has a real eigenvalue. Sum-of-squares rational lower bound certificates were introduced in [12] to overcome the high algebraic degree in the exact real algebraic minima.

2. RADIUS OF POSITIVE SEMIDEFINITENESS

Definition 1 Let $w \in \mathbb{R}_{>0}^n$ be a vector of positive weights. For $x = [x_1, \dots, x_n]^T \in \mathbb{R}^n$ the weighted ℓ^2 norm is

$$\|x\|_{2,w} = \sqrt{w_1 x_1^2 + \dots + w_n x_n^2}.$$

Definition 2 Let $\alpha = [\alpha_1, \dots, \alpha_n] \in \mathbb{R}^n$ be a prescribed real root and $w \in \mathbb{R}_{>0}^n$ a weight vector. The distance to the nearest polynomial with a real root α is defined as

$$\mathcal{N}_{2,w}^{[f]}(\alpha) = \left. \begin{array}{l} \inf_{\tilde{f} \in \mathbb{R}[x_1, \dots, x_n]} \|f - \tilde{f}\|_{2,w}^2 \\ \text{s. t. } \tilde{f}(\alpha) = 0, \\ \deg(\tilde{f}) \leq \deg(f). \end{array} \right\} \quad (1)$$

If f and the used norm is clear from the context, we may write $\mathcal{N}(\alpha)$ for the above infimum, which is actually a minimum (see Theorem 1 below).

Theorem 1 Let $f \in \mathbb{R}[x_1, \dots, x_n]$,

$$f(x_1, \dots, x_n) = \sum_{i_1 + \dots + i_n = 0}^d f_{i_1, \dots, i_n} x_1^{i_1} \dots x_n^{i_n}.$$

For $\tau = [1, \alpha_1, \dots, \alpha_n, \dots, \alpha_1^{i_1} \dots \alpha_n^{i_n}, \dots]^T$, the vector of possible term values in $\tilde{f}$, the distance to the nearest polynomial with a real root α is

$$\mathcal{N}_{2,w}^{[f]}(\alpha) = \frac{f(\alpha)^2}{\tau^T D_w^{-1} \tau}. \quad (2)$$

Furthermore, the coefficient vector $\vec{\tilde{f}}$, for the polynomial $\tilde{f}$ as in (1) is

$$\vec{\tilde{f}} = \vec{f} - \frac{\tau^T \vec{f}}{\tau^T D_w^{-1} \tau} D_w^{-1} \tau, \quad (3)$$

where $\vec{f}$ is the coefficient vector of f and D_w is a diagonal matrix of the weights. The polynomial $\tilde{f}$ is the only polynomial that attains the infimum (2).

Remark 1 The infimum

$$\rho_{2,w}(f) = \inf_{\alpha \in \mathbb{R}^n} \mathcal{N}_{2,w}^{[f]}(\alpha) \quad (4)$$

is the unconstrained radius of positive semidefiniteness. Within any ϵ of the radius (4) there is a polynomial that attains negative values: for any $\epsilon > 0$ there is an $\tilde{f}_\epsilon$ with a real root α and $\|f - \tilde{f}_\epsilon\|_{2,w}^2 < \rho_{2,w}(f) + \epsilon/2$. Then $(\tilde{f}_\epsilon - \delta)(\alpha) < 0$ for all $\delta > 0$, and in particular if $w_1 \delta^2 < \epsilon/2$ we have

$$\|f - (\tilde{f}_\epsilon - \delta)\|_{2,w}^2 < \rho_{2,w}(f) + \epsilon.$$

In Section 4 we will permit constraints for the coefficients of $\tilde{f}$. Then a negative evaluation may be impossible: e.g., $\tilde{f}(x, y) = f_{2,0}x^2 + f_{0,2}y^2$ and $f_{2,0} \geq 0$, $f_{0,2} \geq 0$. However, within less of the distance to the nearest polynomial with a real root a deformed $\tilde{f}$ remains positive definite.

Remark 2 If the weighted norm is the Euclidean norm then the formula becomes

$$\mathcal{N}_2(\alpha) = \frac{f(\alpha)^2}{\sum_{i_1 + \dots + i_n = 0}^d \alpha_1^{2i_1} \dots \alpha_n^{2i_n}}. \quad (5)$$

Remark 3 Different degree conditions in (5) give different rational functions. For example, if the individual variable degrees are bounded by d , $\deg_{x_j}(f) \leq d$ for all j with $1 \leq j \leq n$, then for the ℓ^2 norm

$$\mathcal{N}_2(\alpha) = \frac{f(\alpha)^2}{\sum_{i_1=0}^d \dots \sum_{i_n=0}^d \alpha_1^{2i_1} \dots \alpha_n^{2i_n}}.$$

Comparing the denominators, we have

$$\sum_{i_1=0}^d \dots \sum_{i_n=0}^d \alpha_1^{2i_1} \dots \alpha_n^{2i_n} \geq \sum_{i_1 + \dots + i_n = 0}^d \alpha_1^{2i_1} \dots \alpha_n^{2i_n},$$

so

$$\begin{aligned} \inf \frac{f(\alpha)^2}{\sum_{i_1=0}^d \dots \sum_{i_n=0}^d \alpha_1^{2i_1} \dots \alpha_n^{2i_n}} \\ \leq \inf \frac{f(\alpha)^2}{\sum_{i_1 + \dots + i_n = 0}^d \alpha_1^{2i_1} \dots \alpha_n^{2i_n}}, \end{aligned}$$

which must be since we optimize over a larger set of $\tilde{f}$.

Remark 4 Theorem 1 can be generalized to a complex root α and real/complex coefficients for f , which is the original setting of [3, 15, 27, 24]. Let $f \in \mathbb{C}[x_1, \dots, x_n]$, τ be as described in Theorem 1 then the distance to the nearest polynomial with root $\alpha \in \mathbb{C}^n$ is

$$\mathcal{N}_2^{[f]}(\alpha) = \frac{(\bar{f}(\bar{\alpha}))(f(\alpha))}{\tau^H \tau}.$$

Here ^{H} denotes the Hermitian transposed and $^-$ complex conjugation. Furthermore, the coefficient vector $\vec{\tilde{f}}$, for the polynomial $\tilde{f}$ as in (1) is

$$\vec{\tilde{f}} = \vec{f} - \frac{\tau^T \vec{f}}{\tau^H \tau} \bar{\tau}.$$

Our proof and possible inclusion of weights for the real and imaginary parts is similar to the proof of Theorem 1.

Remark 5 Theorem 1 is the real case of the theorems in [24] and [21] for complex roots. However, they use generalized Lagrangian interpolation for their proof. They also allow keeping selected coefficients of f as their input values and only deform the others in $\tilde{f}$, thus preserving sparsity or monicity, for instance. Our Theorem 1 has theirs as an immediate corollary by setting the weights of those coefficients to ∞ in the limit. However, the problem may become ill-posed. If f has a nonzero constant coefficient which is fixed, and $\alpha = 0$, the set of $\tilde{f}$ is empty. In Section 4 we generalize our approach to handle arbitrary linear constraints on the coefficients of $\tilde{f}$.

If a weight $w_i \rightarrow 0$ in the limit then the corresponding coefficient in $\tilde{f}$ becomes a “don’t care” deformation, i.e., any change in that coefficient is not taken into account in the distance measure. The “nearest” polynomial $\tilde{f}$ with $\alpha \in (\mathbb{R} \setminus \{0\})^n$ as a root then has distance 0, namely $\tilde{f}(x) = f(x) - (f(\alpha)/\alpha^i)x^i$, unless there are additional constraints on the coefficients of $\tilde{f}$ in effect.

Remark 6 The formulas in [7] and [15] use the weights w_i in the denominator of (2), not correctly their reciprocals $1/w_i$.

PROOF OF THEOREM 1. Let $\vec{f}$, τ , and f be as above. Denote the coefficients of $\tilde{f}$ in (1) by

$$\tilde{f}(x_1, \dots, x_n) = \sum_{i_1 + \dots + i_n = 0}^d \tilde{f}_{i_1, \dots, i_n} x_1^{i_1} \dots x_n^{i_n}.$$

Let $\vec{f}$ be the coefficient vector of $\tilde{f}$. Also, $\tilde{f}(\alpha_1, \dots, \alpha_n) = \tau^T \vec{f} = 0$. We have $\|f - \tilde{f}\|_{2,w} = (\vec{f} - \tilde{f})^T D_w (\vec{f} - \tilde{f})$, the weighted ℓ^2 norm, where D_w is a diagonal matrix of the weights. Let λ be the Lagrange multiplier and

$$L(\alpha_1, \dots, \alpha_n, \lambda) = (\vec{f} - \tilde{f})^T D_w (\vec{f} - \tilde{f}) + \lambda \tau^T \vec{f}$$

the Lagrange function of our constrained optimization problem. We must check that α is a regular point (i.e., the gradient of the constraint is not 0 at α). Since $\nabla \tilde{f}(\alpha) \neq 0$ if $\tau \neq 0$ then α is a regular point as long as $\alpha \neq 0$. In the case $\alpha = 0$ the constant coefficient of f is deformed to 0 and the formulas hold. The Jacobian of L w.r.t. $\vec{f}$ and λ is

$$J_L = \begin{bmatrix} \frac{\partial L}{\partial (\vec{f})_i} \\ \frac{\partial L}{\partial \lambda} \end{bmatrix} = \begin{bmatrix} -2D_w(\vec{f} - \tilde{f}) + \lambda \tau \\ \tau^T \vec{f} \end{bmatrix}.$$

Looking at the first block of the vector we have

$$-2D_w(\vec{f} - \tilde{f}) + \tau \lambda = -2D_w \vec{f} + 2D_w \tilde{f} + \tau \lambda = 0. \quad (6)$$

Multiplying by $\tau^T D_w^{-1}$ we have

$$-2\tau^T D_w^{-1} D_w \vec{f} + 2\tau^T D_w^{-1} D_w \tilde{f} + \tau^T D_w^{-1} \tau \lambda = 0.$$

Recalling that $\tilde{f}(\alpha_1, \dots, \alpha_n) = 0$ which means that $\tau^T \tilde{f} = 0$ then we have

$$-2\tau^T D_w^{-1} D_w \vec{f} + \tau^T D_w^{-1} \tau \lambda = -2\tau^T \vec{f} + \tau^T D_w^{-1} \tau \lambda = 0.$$

Solving for λ we get $\lambda = \frac{2\tau^T \vec{f}}{\tau^T D_w^{-1} \tau}$. Looking at (6), we have $\vec{f} - \tilde{f} = \frac{D_w^{-1} \tau \lambda}{2}$. Substituting in for λ we obtain as the only solution $\vec{f} - \tilde{f} = \frac{\tau^T \vec{f}}{\tau^T D_w^{-1} \tau} D_w^{-1} \tau$. Finally,

$$\begin{aligned} \mathcal{N}_{2,w}^{[f]}(\alpha) &= \left(\frac{\tau^T \vec{f}}{\tau^T D_w^{-1} \tau} D_w^{-1} \tau \right)^T D_w \left(\frac{\tau^T \vec{f}}{\tau^T D_w^{-1} \tau} D_w^{-1} \tau \right) \\ &= \frac{\vec{f}^T \tau \tau^T D_w^{-1} \tau \tau^T \vec{f}}{\tau^T D_w^{-1} \tau \tau^T D_w^{-1} \tau}. \end{aligned}$$

$$\text{So } \mathcal{N}_{2,w}^{[f]}(\alpha) = \frac{f(\alpha)^2}{\tau^T D_w^{-1} \tau}. \quad \square$$

Example 1 Here we give another example for the case that the infimum in (4) is not always attainable. Our first example was given at the end of Section 1.1. Consider the polynomial

$$f(x, y) = 1 - 2xy + x^2 y^2 + x^2 = (1 - xy)^2 + x^2.$$

We have that

$$\mathcal{N}_2(\alpha, \beta) = \frac{((1 - \alpha\beta)^2 + \alpha^2)^2}{\sum_{i+j=0}^4 \alpha^{2i} \beta^{2j}}.$$

Then $\inf_{\alpha, \beta} \mathcal{N}_2(\alpha, \beta) = 0$. Suppose now that there exists α, β such that the numerator is 0. Then $(1 - \alpha\beta) = 0$ and $\alpha = 0$. But if $\alpha = 0$ then $\alpha\beta = 0$. Then $1 - \alpha\beta \neq 0$. Contradiction. Thus f does not have a real root and the infimum is not attainable.

We have

$$\mathcal{N}_2(\epsilon, \frac{1}{\epsilon}) = \frac{\epsilon^4}{\delta}, \delta = 3 + 2\epsilon^2 + \frac{2}{\epsilon^2} + 2\epsilon^4 + \frac{2}{\epsilon^4} + \epsilon^6 + \frac{1}{\epsilon^6} + \epsilon^8 + \frac{1}{\epsilon^8},$$

and the nearest polynomial to f with $(\alpha, \beta) = (\epsilon, 1/\epsilon)$ as its root is

$$\begin{aligned} \tilde{f}(x, y) &= -\frac{\epsilon^6}{\delta} x^4 - \frac{\epsilon^4}{\delta} x^3 y + (1 - \frac{\epsilon^2}{\delta}) x^2 y^2 - \frac{1}{\delta} x y^3 - \frac{1}{\epsilon^2 \delta} y^4 \\ &\quad - \frac{\epsilon^5}{\delta} x^3 - \frac{\epsilon^3}{\delta} x^2 y - \frac{\epsilon}{\delta} x y^2 - \frac{1}{\epsilon \delta} y^3 + (1 - \frac{\epsilon^4}{\delta}) x^2 \\ &\quad - (2 + \frac{\epsilon^2}{\delta}) x y - \frac{1}{\delta} y^2 - \frac{\epsilon^3}{\delta} x - \frac{\epsilon}{\delta} y + 1 - \frac{\epsilon^2}{\delta}. \end{aligned}$$

Note that $f(\epsilon, 1/\epsilon) - \epsilon^2 = 0$ has squared distance ϵ^4 from f , which is larger than $\epsilon^4/3 > \epsilon^4/\delta$ for all $\epsilon \neq 0$. $\square$

3. INFINITY AND ONE NORM

Theorem 1 can be generalized to include the ℓ^1 -norm, ℓ^∞ -norm, and any ℓ^p -norm. We discuss in more detail the results presented in [27, 4].

Definition 3 We consider $\mathbb{C}^n$ equipped with some norm $\|\dots\|$. The associated dual norm or operator norm $\|\dots\|^*$ for the column vector $v \in \mathbb{C}^n$ is defined by

$$\|v^T\|^* = \sup_{u \neq 0} \frac{|v^T u|}{\|u\|} = \sup_{\|u\|=1} |v^T u|.$$

Since we are taking the supremum over a compact domain, the maximum value is attained.

Proposition 1 (Proposition 1 in [27]) For each $u \in \mathbb{C}^n$, with $\|u\| = 1$, there exist vectors $v \in \mathbb{C}^n$, with $\|v^T\|^* = 1$, such that $|v^T u| = 1$. $\square$

It is well known that with $\frac{1}{p} + \frac{1}{q} = 1$, $1 \leq p, q \leq \infty$,

$$\|\dots\| = \ell^p\text{-norm} \Leftrightarrow \|\dots\|^* = \ell^q\text{-norm}.$$

Theorem 2 (see [27]) Let $\alpha = [\alpha_1, \dots, \alpha_n] \in \mathbb{C}^n$, $\tau = [1, \alpha_1, \dots, \alpha_n, \dots, \alpha_1^{i_1} \dots \alpha_n^{i_n}, \dots]$, the vector of possible term values of f with norm $\|\dots\|$, and $\vec{f}, \tilde{f} \in \mathbb{C}^n$, with dual norm $\|\dots\|^*$. $\tilde{f}(\alpha) = \tau^T \vec{f} = 0$ requires

$$\|\vec{f} - \tilde{f}\|^* \geq \frac{|f(\alpha)|}{\|\tau\|}.$$

Theorem 2 shows that Theorem 1 can be extended to any ℓ^p -norm. We extend the results from Theorem 2 to the weighted ℓ^1 and ℓ^∞ -norms. We prove Hölder's inequality for weighted ℓ^1 , ℓ^2 and ℓ^∞ -norms, which allows us to then follow the same proof as in [27] for Theorem 2. We further give an explicit formula for $\tilde{f}$.

Theorem 3 Let $u, v \in \mathbb{C}^n$ and weights w_i . Then $|v^T u| \leq \|u\|_{\infty, w} \|v\|_{1, 1/w}$, where $1/w$ is the vector of reciprocals of entries of w .

PROOF. Looking at $|v^T u|$:

$$\begin{aligned} |v^T u| &= \left| \sum_i v_i u_i \right| = \left| \sum_i w_i v_i \frac{1}{w_i} u_i \right| \\ &\leq (\max_j w_j |v_j|) \sum_i \frac{1}{w_i} |u_i| = \|v\|_{\infty, w} \|u\|_{1, 1/w}. \quad \square \end{aligned}$$

Corollary 1 Let $u, v \in \mathbb{C}^n$, and weights w_i . Then

$$|v^T u| \leq \|u\|_{1, w} \|v\|_{\infty, 1/w}.$$

Theorem 4 Let $u, v \in \mathbb{C}^n$ and weights w_i . Then $|v^T u| \leq \|u\|_{2, w} \|v\|_{2, 1/w}$

PROOF OF THEOREM 4. Let $\hat{u}_i = \sqrt{w_i} u_i$, $\hat{v}_i = \frac{v_i}{\sqrt{w_i}}$. Using the Cauchy-Schwartz inequality, we have:
 $|v^T u| = |\hat{v}^T \hat{u}| \leq \left(\sum_i (\sqrt{w_i} u_i)^2 \right)^{1/2} \left(\sum_i \left(\frac{v_i}{\sqrt{w_i}} \right)^2 \right)^{1/2}$.
Therefore, $|v^T u| \leq \|u\|_{2, w} \|v\|_{2, 1/w}$. $\square$

Now that we have proven Hölder's inequality for the weighted ℓ^1 , ℓ^2 and ℓ^∞ -norms, we can follow the proof of Theorem 2 to extend Theorem 1 to weighted ℓ^1 and ℓ^∞ -norms. Theorem 4 would also yield an alternative proof of Theorem 1.

Theorem 5 For τ, f , and $\tilde{f}$ as described in Theorem 2, $v = [1, \text{sgn}(\tau_i), \dots]$, where $v \in \mathbb{R}^\kappa$, κ is the dimension of f and $\text{sgn}(\tau_i) = 1$ if $\tau_i > 0$, $\text{sgn}(\tau_i) = -1$ if $\tau_i < 0$, and $\text{sgn}(\tau_i) = 0$ if $\tau_i = 0$ with weighted ℓ^∞ -norm and weights w_i then

$$\mathcal{N}_{\infty, w}^{[f]}(\alpha) = \frac{|f(\alpha)|}{\|\tau\|_{1, 1/w}} \text{ and } \vec{f} = \tilde{f} - \frac{f(\alpha)}{\|\tau\|_{1, 1/w}} D_w^{-1} v.$$

PROOF OF THEOREM 5. From [27] and Theorem 3 we know that $|f(\alpha)| = |(\vec{f} - \tilde{f})^T \tau| \leq \|\vec{f} - \tilde{f}\|_{\infty, w} \|\tau\|_{1, 1/w}$. Therefore, $\frac{|f(\alpha)|}{\|\tau\|_{1, 1/w}} \leq \|\vec{f} - \tilde{f}\|_{\infty, w}$. For all j choose $(\tilde{f})_j$ such that $\frac{f(\alpha)}{\|\tau\|_{1, 1/w}} = w_j(\vec{f} - \tilde{f})_j$. From this we

get that $w_j(\tilde{f})_j = w_j(\vec{f})_j - \frac{f(\alpha)}{\|\tau\|_{1, 1/w}}$. Therefore, $\tilde{f} = \vec{f} - \frac{f(\alpha)}{\|\tau\|_{1, 1/w}} D_w^{-1} v$, which yields equality in the above inequality. This gives $0 = \tilde{f}(\alpha) = \tau^T \vec{f} - \frac{f(\alpha)}{\|\tau\|_{1, 1/w}} \tau^T D_w^{-1} v$. $\square$

In the same way we obtain the following theorem.

Theorem 6 For $\tau, f, \tilde{f}$, and $\text{sgn}(\tau_i)$ as described in Theorem 5 with weighted ℓ^1 -norm and weights $w_i \geq 0$ then

$$\mathcal{N}_{1, w}^{[f]}(\alpha) = \frac{|f(\alpha)|}{\|\tau\|_{\infty, 1/w}} \text{ and}$$

$$\vec{f}_i = \begin{cases} \tilde{f}_i & \text{for } i \neq j \\ \tilde{f}_i - \text{sgn}(\tau_i) \frac{f(\alpha)}{\|\tau\|_{\infty, 1/w}} \frac{1}{w_i} & \text{for } i = j \end{cases}$$

$$\text{where } \frac{|\tau_j|}{w_j} = \max_i \frac{|\tau_i|}{w_i}.$$

4. GENERALIZATIONS

Our method can be further generalized to include problems with linear constraints of the form $H \vec{f} = p$, where $H \in \mathbb{R}^{t \times s}$, $p \in \mathbb{R}^t$, on the coefficient vector $\vec{f}$ of $\tilde{f}$. We define

$$\mathcal{N}_{2, w}^{[f; H]}(\alpha) = \left. \begin{aligned} &\inf_{\tilde{f} \in \mathbb{R}[x_1, \dots, x_n]} \|f - \tilde{f}\|_{2, w}^2 \\ &\text{s. t. } \tilde{f}(\alpha) = 0, H \vec{f} = p, \\ &\quad \deg(\tilde{f}) \leq \deg(f). \end{aligned} \right\} \quad (7)$$

We note that the Jacobian of the Lagrange function corresponding to (7) constitutes a linear system in the unknown coefficients of $\tilde{f}$ and the multipliers, hence a determinantal formula parameterized by the real root for the solution can be computed, which one can minimize.

Example 2 Given a polynomial $f(x, y) = x^2 + y^2 + 1$ find the nearest polynomial $\tilde{f}(x, y) = \tilde{f}_{2,0} x^2 + \tilde{f}_{0,2} y^2 + \tilde{f}_{1,1} xy + \tilde{f}_{1,0} x + \tilde{f}_{0,1} y + \tilde{f}_{0,0}$ with $\tilde{f}_{1,1} = \tilde{f}_{0,0}$ and $\tilde{f}_{0,1} = 0$ and with the root (α, β) . The Lagrangian function is

$$\begin{aligned} L(\alpha, \beta, \lambda) &= (\vec{f} - \tilde{f})^T (\vec{f} - \tilde{f}) \\ &\quad + \lambda_0 \tau^T \vec{f} + \lambda_1 (\tilde{f}_{1,1} - \tilde{f}_{0,0}) + \lambda_2 \tilde{f}_{0,1}, \end{aligned}$$

where the term vector $\tau = [\alpha^2, \beta^2, \alpha\beta, \alpha, \beta, 1]$. The Jacobian of L in $\vec{f}$ and λ is zero for

$$\vec{f} = \begin{bmatrix} \tilde{f}_{2,0} \\ \tilde{f}_{0,2} \\ \tilde{f}_{1,1} \\ \tilde{f}_{1,0} \\ \tilde{f}_{0,1} \\ \tilde{f}_{0,0} \end{bmatrix} = \begin{bmatrix} -\frac{\alpha^2 - 2\beta^4 + \alpha^2 \beta^2 - 2\alpha\beta - 1 + \alpha^3 \beta}{2\alpha^2 + 2\beta^4 + 2\alpha^4 + \alpha^2 \beta^2 + 2\alpha\beta + 1} \\ \frac{2\alpha^2 + 2\beta^4 - \alpha^2 \beta^2 + 2\alpha\beta + 1 - \beta^2 - \alpha\beta^3}{2\alpha^2 + 2\beta^4 + 2\alpha^4 + \alpha^2 \beta^2 + 2\alpha\beta + 1} \\ \frac{\beta^4 + \alpha^4 - \alpha\beta^3 - \alpha^3 \beta - \beta^2}{2\alpha^2 + 2\beta^4 + 2\alpha^4 + \alpha^2 \beta^2 + 2\alpha\beta + 1} \\ \frac{\alpha(1 + 2\beta^2 + \alpha\beta + 2\alpha^2)}{2\alpha^2 + 2\beta^4 + 2\alpha^4 + \alpha^2 \beta^2 + 2\alpha\beta + 1} \\ 0 \\ \frac{\beta^4 + \alpha^4 - \alpha\beta^3 - \alpha^3 \beta - \beta^2}{2\alpha^2 + 2\beta^4 + 2\alpha^4 + \alpha^2 \beta^2 + 2\alpha\beta + 1} \end{bmatrix},$$

$$\lambda_0 = \frac{2(2\alpha^2 + \alpha\beta + 2\beta^2 + 1)}{2\alpha^4 + 2\alpha^2 + \alpha^2 \beta^2 + 2\alpha\beta + 1 + 2\beta^4},$$

$$\lambda_1 = \frac{-2(\alpha^4 + \alpha^3 \beta + \alpha^2 \beta^2 + \alpha\beta + \beta^3 \alpha + \beta^4 - \beta^2)}{2\alpha^4 + 2\alpha^2 + \alpha^2 \beta^2 + 2\alpha\beta + 1 + 2\beta^4},$$

$$\lambda_2 = \frac{-2(2\alpha^2 + \alpha\beta + 2\beta^2 + 1)\beta}{(2\alpha^4 + 2\alpha^2 + \alpha^2\beta^2 + 2\alpha\beta + 1 + 2\beta^4)}.$$

The minimum perturbation is

$$\mathcal{N}_2 = \frac{3\alpha^4 + 2\alpha^3\beta + 5\alpha^2\beta^2 + 3\alpha^2 + 2\alpha\beta + 2\alpha\beta^3 + 1 + 3\beta^4 + 2\beta^2}{2\alpha^2 + 2\alpha^4 + 2\beta^4 + \alpha^2\beta^2 + 2\alpha\beta + 1} \quad (8)$$

Running the **Minimize** procedure in Maple 13 we obtain $\min_{(\alpha,\beta)} \mathcal{N}_2 = 1$ at the root $(0,0)$ and $\tilde{f} = x^2 + y^2$. That is the same deformed polynomial as for the unconstrained problem but derived from a different norm expression (8).

Note that before minimizing (8) one could restrict (α, β) to lie on a parametric curve, thus constraining the variables rather than the coefficients, as is done in [7]. $\square$

Our method can be generalized even further to include inequalities, $G\tilde{f} \leq q$ with $G \in \mathbb{R}^{m \times s}$. Then

$$\mathcal{N}_{2,w}^{[f;H;G]}(\alpha) = \left. \begin{array}{l} \inf_{\tilde{f} \in \mathbb{R}[x_1, \dots, x_n]} \|f - \tilde{f}\|_{2,w}^2 \\ \text{s. t. } \tilde{f}(\alpha) = 0, G\tilde{f} \leq q, H\tilde{f} = p, \\ \deg(\tilde{f}) \leq \deg(f). \end{array} \right\} \quad (9)$$

Note that our constraint functions, being linear, are always convex. We can use the Karush-Kuhn-Tucker (KKT) conditions and the quantities as defined in Theorem 1. Using the KKT conditions in equation (9) with the Lagrange function

$$L = (\tilde{f} - \tilde{f})^T D_w(\tilde{f} - \tilde{f}) + \lambda_0 \tau^T \tilde{f} + \lambda^T (H\tilde{f} - p) + \mu^T (G\tilde{f} - q).$$

The KKT conditions (for a regular point) are then

$$\left. \begin{array}{l} \frac{\partial L}{\partial(\tilde{f})_i} = 0, \quad i = 1, \dots, s, \\ \tau^T \tilde{f} = 0, \\ H\tilde{f} = p, \\ G\tilde{f} \leq q, \\ \mu_i \geq 0, \quad i = 1, \dots, m, \\ \mu^T (G\tilde{f} - q) = 0. \end{array} \right\} \quad (10)$$

The last orthogonality conditions constitute branching: $\mu_i = 0$ or $(G\tilde{f} - q)_i = 0$, and (10) form linear programs.

Example 3 Given a polynomial $f(x, y) = x^2 + y^2 - 2y + 1$ and constraint $\tilde{f}_{0,1} \geq 0$, we determine the nearest polynomial $\tilde{f}(x, y) = \tilde{f}_{2,0}x^2 + \tilde{f}_{0,2}y^2 + \tilde{f}_{1,1}xy + \tilde{f}_{1,0}x + \tilde{f}_{0,1}y + \tilde{f}_{0,0}$ with real root $(0,0)$. The term vector for the root is $\tau = [0, 0, 0, 0, 0, 1]$. The Lagrangian function is $L(\alpha, \beta, \lambda, \mu) = (\tilde{f} - \tilde{f})^T (\tilde{f} - \tilde{f}) + \lambda \tau^T \tilde{f} + \mu(-\tilde{f}_{0,1})$. We can formulate the KKT conditions as solving two linear programs:

$$\begin{array}{ll} \text{Minimize} & 1 \\ \text{subject to} & \partial L / \partial(\tilde{f})_i = 0, \quad i = 1, \dots, 6, \\ & \tilde{f}_{0,0} = 0, \\ & -\tilde{f}_{0,1} \leq 0, \\ & \mu = 0, \end{array}$$

and

$$\begin{array}{ll} \text{Minimize} & 1 \\ \text{subject to} & \partial L / \partial(\tilde{f})_i = 0, \quad i = 1, \dots, 6, \\ & \tilde{f}_{0,0} = 0, \\ & -\tilde{f}_{0,1} = 0, \\ & \mu \geq 0. \end{array}$$

The first linear program is infeasible; for the second linear program we obtain:

$$\tilde{f} = x^2 + y^2, \lambda = 2, \mu = 4, \text{ and } \mathcal{N}_2^{[f;G]} = 5.$$

The minimum perturbation can also be obtained by running the **Minimize** procedure in Maple 13 on the original optimization problem (9). $\square$

The above result can be extended to systems. The distance to the nearest system with k equations and common root α is defined as

$$\left. \begin{array}{l} \inf_{\tilde{f}_1, \dots, \tilde{f}_k} \|f_1 - \tilde{f}_1\|_2^2 + \dots + \|f_k - \tilde{f}_k\|_2^2 \\ \text{s. t. } \tilde{f}_i(\alpha) = 0, \quad i = 1, \dots, k \\ \tilde{f}_i \in \mathbb{R}[x_1, \dots, x_n], \quad i = 1, \dots, k \\ \deg(\tilde{f}_i) \leq \deg(f_i), \quad i = 1, \dots, k \end{array} \right\} \quad (11)$$

Applying Theorem 1 and Theorem 5 to each individual $\tilde{f}_k$ easily yields the following.

Theorem 7 Let $f_1, \dots, f_k \in \mathbb{R}[x_1, \dots, x_n]$, with $d_i = \deg(f_i)$. The distance to the nearest system with a common root $\alpha \in \mathbb{R}^n$ is in ℓ^2 -norm

$$\mathcal{N}_2^{\{f_1, \dots, f_k\}}(\alpha) = \frac{f_1(\alpha)^2}{\sum_{i_1 + \dots + i_n = 0}^{d_1} \alpha_1^{2i_1} \dots \alpha_n^{2i_n}} + \dots + \frac{f_k(\alpha)^2}{\sum_{i_1 + \dots + i_n = 0}^{d_k} \alpha_1^{2i_1} \dots \alpha_n^{2i_n}}, \quad (12)$$

and in ℓ^∞ -norm

$$\mathcal{N}_{\infty, w}^{\{f_1, \dots, f_k\}}(\alpha) = \max_{1 \leq j \leq k} \frac{|f_j(\alpha)|}{\|\tau\|_{1,1/w}}.$$

The nearest polynomials, if they exist (see Example 1) are again determined by (3). Theorem 7 easily generalizes to include weighted norms. Linear equality and inequality constraints on the coefficients as described in (7) and (9) can also be applied.

Example 4 Given polynomials

$$f_1(x, y) = x^4 + y^4 + 1 \text{ and } f_2(x, y) = x^2 + x^2 y^2 - 2xy + 1$$

we shall determine the minimum perturbation such that the deformed system of 2 equations has a real root.

For that, we compute the Gröbner basis of the numerators of the partial derivatives of (12) (cf. [2]). In Section 5 we present an alternative approach based on sum-of-squares certificates. The first equation in the obtained Gröbner basis is a polynomial in terms of β of degree 195. Next, we find all real roots of this polynomial and plug all 9 choices into a second polynomial in the Gröbner basis. We compute the norm of each possible point and select the minimum value. The minimum

perturbation obtained by solving the Gröbner basis of (12) in Maple is

$$\tilde{\mathcal{N}}_2 = 0.64597306998078277667 \quad (13)$$

for

$$(\alpha, \beta) = (-0.9138289555225176138, -1.1947071766554875688).$$

Note that for this example at least 25 mantissa digits must be used in Maple 13 in order to obtain the correct minimum.

We can then find the nearest polynomial system by plugging the root into equation (3) for each of the two polynomials:

$$\begin{aligned} \tilde{f}_1 &= 0.83448994938 + 0.15028000318x + \\ &0.19773604528y - 0.17954059831xy - \\ &0.13645140747x^2 - 0.23623667238y^2 + \\ &0.12389530347x^3 + 0.21449844130xy^2 + \\ &0.16301947576x^2y + 0.28223364788y^3 + \\ &0.88750540206x^4 - 0.14801860821x^3y - \\ &0.19476053763x^2y^2 - 0.25626282720xy^3 + \\ &0.66281343538y^4, \\ \tilde{f}_2 &= 0.96296934167 + 0.03362313909x + \\ &0.04424079327y - 2.04016980557xy + \\ &0.96947082410x^2 - 0.05285479322y^2 + \\ &0.02771991571x^3 + 0.04799115499xy^2 + \\ &0.03647342555x^2y + 0.06314600078y^3 - \\ &0.02516916045x^4 - 0.03311718223x^3y + \\ &0.95642493674x^2y^2 - 0.05733537729xy^3 - \\ &0.07544098031y^4. \quad \square \end{aligned}$$

5. LOWER BOUND CERTIFICATES

The minimization of the rational function $\mathcal{N}_{2,w}^{[f]} = \frac{f(\alpha)^2}{g(\alpha)^2}$ where $g = \tau^T D_w^{-1} \tau$ defined in (2) can be reformulated as maximizing r such that $f(\alpha)^2 - rg(\alpha)$ is nonnegative. We compute a lower bound of $\inf_{\alpha \in \mathbb{R}^n} \mathcal{N}_{2,w}^{[f]}(\alpha)$ by solving the SOS program [10, 19, 12, 14]:

$$\left. \begin{aligned} r^* &:= \sup_{r \in \mathbb{R}, W} r \\ \text{s. t. } &f(\mathbf{X})^2 - rg(\mathbf{X}) = m_d(\mathbf{X})^T W m_d(\mathbf{X}) \\ &W \succeq 0, W^T = W \end{aligned} \right\} \quad (14)$$

where $m_d(\mathbf{X})$ is the column vector of all terms in $\mathbf{X}_1, \dots, \mathbf{X}_n$ up to degree d . The dimension of $m_d(\mathbf{X})$ is $\binom{n+d}{d}$.

The SOS program (14) can be solved efficiently by algorithms in GloptiPoly [6], SOSTOOLS [22], YALMIP [17] and SeDuMi [28]. One can use GloptiPoly as described in [6] to extract the solutions α which achieve the global minimum. However, since we are running fixed precision SDP solvers in Matlab, we can only obtain a numerical positive semidefinite matrix W and floating point number r^* which satisfy approximately

$$f(\mathbf{X})^2 - r^*g(\mathbf{X}) \approx m_d(\mathbf{X})^T \cdot W \cdot m_d(\mathbf{X}), \quad W \succeq 0. \quad (15)$$

So r^* is a lower bound of $\inf_{\alpha \in \mathbb{R}^n} \mathcal{N}_{2,w}^{[f]}(\alpha)$ approximately!

The lower bound $\tilde{r}$ is certified if $\tilde{r}$ and $\tilde{W}$ hold the following conditions exactly:

$$f(\mathbf{X})^2 - \tilde{r}g(\mathbf{X}) = m_d(\mathbf{X})^T \cdot \tilde{W} \cdot m_d(\mathbf{X}), \quad \tilde{W} \succeq 0. \quad (16)$$

We can use Artin's theorem of sum-of-squares and semidefinite programming (see, e.g., [20, 12, 14]) to certify the computed minimum. We have done so for the minimum 1 of (8) of Example 2 and the rational lower bound $\tilde{\mathcal{N}}_2 = 64597306998078108/10000000000000000$ of the real algebraic optimum (13) of Example 4.

Example 5 ([29]) Given a polynomial

$$\begin{aligned} f &= x^2y^2 + x^2 - xy + y^4 - y^2 + 1 \\ &= (xy - 1/2)^2 + (y^2 - 1/2)^2 + x^2 + 1/2, \end{aligned}$$

decide the minimum perturbation such that the perturbed polynomial has a real root.

If we allow dense perturbations, after running `solvesos` in Matlab, we get the lower bound

$$\tilde{\mathcal{N}}_2 = 2.453484553428391600 \times 10^{-15}.$$

This is caused by the assumption that we can perturb f by any monomial terms with degree bounded by 4. In general, for $f(x, y) - \epsilon x^4$ one has for $x = y^2$ that $g(y) = f(y^2, y) - \epsilon y^8$. Notice that $g(y)$ always has a real root, because $g(0) = 1$ and $g(\infty) = -\infty$. We see that f has a radius of positive semidefiniteness that is 0. Hence, it would be more interesting to consider a weighted norm. For instance, if we only allow terms which appear in f to be perturbed, then the lower bound computed by `solvesos` in Matlab is $\tilde{\mathcal{N}}_2 = 0.2469160193369205900$. After applying the certification algorithm in [12, 14], we obtain the certified lower bound

$$\tilde{\mathcal{N}}_2 = 24691601933692029/100000000000000000. \quad \square$$

This means that f is positive since $f(0, 0) = 1 > 0$.

Example 6 (see [18]) Consider the polynomial

$$f(x, y) = 2 - 3x^2y^2 + x^2y^4 + x^4y^2.$$

Notice that f is the result of adding one to the Motzkin polynomial. It is well-known that f is positive semidefinite but not an SOS, as seen in [18]. In fact $f \geq 1$ for all $x, y \in \mathbb{R}$. First, we consider using a dense perturbation to obtain a lower bound for $\mathcal{N}_2$. We use Matlab to compute the approximate lower bound of $\mathcal{N}_2$ and obtain 0 as the minimum, which is easily proven by considering $f(x, y) - \epsilon x^5$. Hence, we consider a weighted norm. We use infinite weights on the terms that have zero coefficients in f . Thus, we only allow the terms which appear in f to be perturbed (sparse deformation). The lower bound computed by `solvesos` in Matlab is

$$\tilde{\mathcal{N}}_2 = 0.1285480262594671800.$$

After applying the certification algorithm in [12, 14], we obtain the certified lower bound

$$\tilde{\mathcal{N}}_2 = 12854802625942833/100000000000000000.$$

We have computed an exact rational certificate (as in (16)) $f(x, y)^2 - 12854802625942833/100000000000000000 \times (1 + x^4y^8 + x^8y^4 + x^4y^4) = \text{SOS}$ (10 polynomial squares).

This means that the non-zero coefficients of f need to be perturbed (by at least 0.128 in ℓ^2 -norm squared) for f to have a real root. Since $f(0, 0) = 2$, we have

proven that $f(x, y) > 0$ for all real x, y via a polynomial sum-of-squares certificate. $\square$

Example 6 answers a question by one of the referees. In fact, we conjecture that such polynomial sums-of-squares always exist. More precisely, if for a real polynomial $f(x_1, \dots, x_n)$ there exists a vector w of positive and infinite weights (excluding an infinite weight for the constant coefficient) such that $\rho_{2,w}(f) > 0$ then in (14) $r^* > 0$. We have seen that $\rho_{2,w}(f)$ easily is no larger than 0, provided f has a projective root at infinity, and the condition $\rho_{2,w}(f) > 0$ makes f and w quite special.

Acknowledgments: Sharon Hutton thanks the EAPSI program staff at NSF and the staff at KLMM for their assistance with her 2009 Summer Institute in Beijing. We thank Mohab Safey El Din for his comments on Seidenberg's problem, and the reviewers for their remarks.

6. REFERENCES

- [1] AUBRY, P., ROULLIER, F., AND SAFEY EL DIN, M. Real solving for positive dimensional systems. *J. Symbolic Comput.* 34, 6 (Dec. 2002), 543–560. URL: <http://www-spiral.lip6.fr/~safey/Articles/RR-3992.ps.gz>.
- [2] BECKER, E., POWERS, V., AND WÖRMANN, T. Deciding positivity of real polynomials. In *Real Algebraic Geometry and Ordered Structures*, C. N. Delzell and J. J. Madden, Eds., vol. 253 of *Contemporary Math.* AMS, 2000, pp. 251–272.
- [3] CORLESS, R. M., GIANNI, P. M., TRAGER, B. M., AND WATT, S. M. The singular value decomposition for polynomial systems. In *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. ISSAC'95* (New York, N. Y., 1995), A. H. M. Levelt, Ed., ACM Press, pp. 96–103.
- [4] CORLESS, R. M., KALTOFEN, E., AND WATT, S. M. Hybrid methods. In *Computer Algebra Handbook*, J. Grabmeier, E. Kaltofen, and V. Weispfenning, Eds. Springer Verlag, Heidelberg, Germany, 2003, Section 2.12.3, pp. 112–125. URL: <http://www.math.ncsu.edu/~kaltofen/bibliography/01/symnum.pdf>.
- [5] ECKER, J. G., AND KUPFERSCHMID, M. *Introduction to Operations Research*. John Wiley and Sons, 1988. 509 pp.
- [6] HENRION, D., AND LASSEIRE, J.-B. Detecting global optimality and extracting solutions in GloptiPoly. In *Positive polynomials in control*, D. Henrion and A. Garulli, Eds., vol. 312 of *Lecture Notes on Control and Information Sciences*. Springer Verlag, Heidelberg, Germany, 2005, pp. 293–310. URL: <http://homepages.laas.fr/henrion/Papers/extract.pdf>.
- [7] HITZ, M. A., AND KALTOFEN, E. Efficient algorithms for computing the nearest polynomial with constrained roots. In *Proc. 1998 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'98)* (New York, N. Y., 1998), O. Gloor, Ed., ACM Press, pp. 236–243. URL: <http://www.math.ncsu.edu/~kaltofen/bibliography/98/HiKa98.pdf>.
- [8] HITZ, M. A., KALTOFEN, E., AND LAKSHMAN Y. N. Efficient algorithms for computing the nearest polynomial with a real root and related problems. In *Proc. 1999 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'99)* (New York, N. Y., 1999), S. Dooley, Ed., ACM Press, pp. 205–212. URL: <http://www.math.ncsu.edu/~kaltofen/bibliography/99/HKL99.pdf>.
- [9] JACOBSON, N. *Basic Algebra I*. W. H. Freeman & Co., San Francisco, 1974.
- [10] JIBETAN, D., AND DE KLERK, E. Global optimization of rational functions: a semidefinite programming approach. *Math. Program.* 106, 1 (2006), 93–109.
- [11] KALTOFEN, E. Computing the irreducible real factors and components of an algebraic curve. *Appl. Algebra Engin. Commun. Comput.* 1, 2 (1990), 135–148. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/90/Ka90_aaecc.pdf.
- [12] KALTOFEN, E., LI, B., YANG, Z., AND ZHI, L. Exact certification of global optimality of approximate factorizations via rationalizing sums-of-squares with floating point scalars. In *ISSAC 2008* (New York, N. Y., 2008), D. Jeffrey, Ed., ACM Press, pp. 155–163. URL: <http://www.math.ncsu.edu/~kaltofen/bibliography/08/KLYZ08.pdf>.
- [13] KALTOFEN, E., YANG, Z., AND ZHI, L. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *ISSAC MMVI Proc. 2006 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2006), J.-G. Dumas, Ed., ACM Press, pp. 169–176. URL: <http://www.math.ncsu.edu/~kaltofen/bibliography/06/KYZ06.pdf>.
- [14] KALTOFEN, E. L., LI, B., YANG, Z., AND ZHI, L. Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients, Jan. 2009. Accepted for publication in *J. Symbolic Comput.* URL: <http://www.math.ncsu.edu/~kaltofen/bibliography/09/KLYZ09.pdf>.
- [15] KARMARKAR, N. K., AND LAKSHMAN Y. N. On approximate GCDs of univariate polynomials. *J. Symbolic Comput.* 26, 6 (1998), 653–666. Special issue on Symbolic Numeric Algebra for Polynomials S. M. Watt and H. J. Stetter, editors.
- [16] KHARITONOV, V. L. Asymptotic stability of an equilibrium of a family of systems of linear differential equations. *Differential Equations* 14 (1979), 1483–1485.
- [17] LÖFBERG, J. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proc. IEEE CCA/ISIC/CACSD Conf.* (Taipei, Taiwan, 2004). URL: <http://control.ee.ethz.ch/~joloef/yalmip.php>.
- [18] MARSHALL, M. *Positive Polynomials and Sums of Squares*. American Math. Soc., 2008. 187 pp.
- [19] NIE, J., DEMMEL, J., AND GU, M. Global minimization of rational functions and the nearest GCDs. *J. of Global Optimization* 40, 4 (2008), 697–718.
- [20] PEYRL, H., AND PARRILO, P. A. Computing sum of squares decompositions with rational coefficients. *Theoretical Comput. Sci.* 409 (2008), 269–281.
- [21] POPE, S., AND SZANTO, A. Nearest multivariate system with given root multiplicities. *Journal of Symbolic Computation* 44 (2009), 606–625. URL: <http://www4.ncsu.edu/~aszanto/Pope-Szanto-Preprint.pdf>.
- [22] PRAJNA, S., PAPACHRISTODOULOU, A., SEILER, P., AND PARRILO, P. A. SOSTOOLS: Sum of squares optimization toolbox for MATLAB. Available from <http://www.cds.caltech.edu/sostools> and <http://www.mit.edu/~parrilo/sostools>, 2004.
- [23] REZVANI, N., AND CORLESS, R. M. The nearest singular polynomial with a given zero, revisited. *SIGSAM Bulletin* 39, 3 (Sept. 2005), 73–79.
- [24] RUATTA, O., SCIABICA, M., AND SZANTO, A. Over-constrained Weierstrass iteration and the nearest consistent system. Rapport de Recherche 5215, INRIA, www.inria.fr, June 2004. Accepted in *Journal of Complexity*. URL: <ftp://ftp-sop.inria.fr/galaad/oruatta/RR-5215.pdf>.
- [25] SAFEY EL DIN, M. *Résolution réelle des systèmes polynomiaux en dimension positive*. Thèse de doctorat, Univ. Paris VI (Univ. Pierre et Marie Curie), Paris, France, 2001. URL: http://www-spiral.lip6.fr/~safey/these_safey.ps.gz.
- [26] SEIDENBERG, A. A new decision method for elementary algebra. *Annals Math.* 60 (1954), 365–374.
- [27] STETTER, H. J. The nearest polynomial with a given zero, and similar problems. *SIGSAM Bulletin* 33, 4 (Dec. 1999), 2–4.
- [28] STURM, J. F. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software* 11/12 (1999), 625–653.
- [29] ZENG, G. Nonstandard decision methods for the solvability of real polynomial equations. *Science in China Series-A Mathematics* 42, 12 (1999), 1251–1261.
- [30] ZHI, L., NODA, M.-T., KAI, H., AND WU, W. Hybrid method for computing the nearest singular polynomials. *Japan J. Industrial and Applied Math.* 21, 2 (June 2004), 149–162. URL: <http://www.mmrc.iss.ac.cn/~lzh/Publications/zknwjiam04.pdf>.
- [31] ZHI, L., AND WU, W. Nearest singular polynomial. *J. Symbolic Comput.* 26, 6 (1998), 667–675. Special issue on Symbolic Numeric Algebra for Polynomials S. M. Watt and H. J. Stetter, editors.

Random Polynomials and Expected Complexity of Bisection Methods for Real Solving

Ioannis Z. Emiris
University of Athens
Athens, Greece
emiris(at)di.uoa.gr

André Galligo
University of Nice
Nice, France
galligo(at)unice.fr

Elias P. Tsigaridas
Århus University, Denmark
University of Athens, Greece
elias.tsigaridas(at)gmail.com

ABSTRACT

Our probabilistic analysis sheds light to the following questions: Why do random polynomials seem to have few, and well separated real roots, on the average? Why do exact algorithms for real root isolation may perform comparatively well or even better than numerical ones?

We exploit results by Kac, and by Edelman and Kostlan in order to estimate the real root separation of degree d polynomials with i.i.d. coefficients that follow two zero-mean normal distributions: for $SO(2)$ polynomials, the i -th coefficient has variance $\binom{d}{i}$, whereas for Weyl polynomials its variance is $1/i!$. By applying results from statistical physics, we obtain the expected (bit) complexity of STURM solver, $\tilde{O}_B(rd^2\tau)$, where r is the number of real roots and τ the maximum coefficient bitsize. Our bounds are two orders of magnitude tighter than the record worst case ones. We also derive an output-sensitive bound in the worst case.

The second part of the paper shows that the expected number of real roots of a degree d polynomial in the Bernstein basis is $\sqrt{2d} \pm \mathcal{O}(1)$, when the coefficients are i.i.d. variables with moderate standard deviation. Our paper concludes with experimental results which corroborate our analysis.

Categories and Subject Descriptors

F.2 [Theory of Computation]: Analysis of Algorithms and Problem Complexity; I.1 [Computing Methodology]: Symbolic and algebraic manipulation Algorithms

General Terms

Algorithms, Theory

Keywords

Random polynomial, real-root isolation, Bernstein polynomial, expected complexity, separation bound

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

1. INTRODUCTION

One of the most important procedures in computer algebra and algebraic algorithms is root isolation of univariate polynomials. The goal is to compute intervals in the real case, or squares in the complex case, that isolate the roots of the polynomial and to compute one such interval, or square, for every root.

We restrict ourselves to exact algorithms, i.e. algorithms that perform arithmetic with rational numbers of arbitrary size. The best known algorithms are subdivision algorithms, based on Sturm sequences (STURM), or on Descartes' rule of sign (DESCARTES), or on Descartes' rule and the Bernstein basis representation (BERNSTEIN). Subdivision algorithms mimic binary search and their complexity depends on separation bounds. They are given an initial interval, or compute one containing all real roots. Then, they repeatedly subdivide it until it is certified that zero or one real root is contained in the tested interval.

Thanks to important recent progress [7, 8, 10, 11], the complexity of STURM, DESCARTES and BERNSTEIN is, in the worst case, $\tilde{O}_B(d^4\tau^2)$, where d is the degree of the polynomial and τ the maximum coefficient bitsize. The bound holds even when the polynomial is non-squarefree, and we also compute (all) the multiplicities. This requires a preprocessing of complexity $\tilde{O}_B(d^2\tau)$, in order to compute the square-free factorization. The new polynomial has coefficients of size $\mathcal{O}(d + \tau)$. The complexity of this stage, although significant in practice, is asymptotically dominated. In this paper we consider the behavior of STURM on random polynomials of various forms. Our results can be extended to DESCARTES and BERNSTEIN.

Another important exact solver (CF) is based on the continued fractions expansion of the real roots e.g. [1, 33, 35]. Several variants of this solver exist, depending on the method used to compute the partial quotients of the real roots. Assuming the Gauss-Kuzmin distribution holds for the real algebraic numbers, it was proven [35], that the expected complexity is $\tilde{O}_B(d^4\tau^2)$. By spreading the roots, the expected complexity becomes $\tilde{O}_B(d^3\tau)$ [35]. The currently known worst-case bound is $\tilde{O}_B(d^4\tau^2)$ [25]. This paper reduces the gap between STURM CF.

Numerical algorithms compute an approximation, up to a desired accuracy, of all complex roots. They can be turned into isolation algorithms by requiring the accuracy to be equal to the theoretical worst-case separation bound. The current record is $\tilde{O}_B(d^3\tau)$ and is achieved by recursively splitting the polynomial until one obtains linear factors that approximate sufficiently the roots [32, 27]. It seems that the

bounds could be improved to $\tilde{\mathcal{O}}_B(d^2\tau)$ with a more sophisticated splitting process. We should mention that optimal numerical algorithms are very difficult to implement.

Even though the complexity bounds of the exact algorithms are worse than those of the numerical ones, recent implementations of the former tend to be competitive, if not superior, in practice, e.g. [19, 30, 11, 35]. Our work attempts to provide an explanation for this. There is a huge amount of work concerning root isolation and the references stated represent only the tip of the iceberg; we encourage the reader to refer to the references.

Most of the work on random polynomials, which typically concerns polynomials in the monomial basis, focuses on the *number* of real roots. Kac’s [20] celebrated result estimated the expected number of real roots of random polynomials (named after himself) as $\frac{2}{\pi} \log d + \mathcal{O}(1)$, when the coefficients are standard normals i.i.d. or uniformly distributed, and d is the degree of the polynomial. We refer the reader to e.g. [5, 24, 12] for a historical perspective and to [3] for various references. A geometric interpretation of this result and many generalizations appear in [9]. We mainly examine $SO(2)$ polynomials, where the i -th coefficient is an i.i.d. Gaussian random variable of zero mean and variance $\binom{d}{i}$. According to [9], they are “the most natural definition of random polynomials”, see also [34]. Their expected number of real roots is $\sqrt{d}$. For Weyl polynomials, the i -th coefficient is an i.i.d. Gaussian random variable of zero mean and variance $1/i!$, and the expected number of real roots is about $\frac{2}{\pi} \sqrt{d} + \mathcal{O}(1)$ where higher-order terms are not known to date [31]. For results on complex roots we refer to e.g. [14, 13].

Our first contribution concerns the expected bit complexity of STURM, when the input is random polynomials with i.i.d. coefficients; notice that their roots are *not* independently distributed! In other words, we have to go beyond the theory of Kac, and Edelman and Kostlan, in order to study the statistical behavior of root differences and, more precisely, the *minimum* absolute difference. We examine $SO(2)$ and Weyl random polynomials, and exploit the relevant progress achieved in statistical physics. In fact, these polynomial classes are of particular interest in statistical physics because they model zero-crossings in diffusion equations and, eventually, a chaotic spin wave-function [4, 14, 31]. The key observation is that, by applying these results, we can quantify the correlation between the roots, which is sufficiently weak, but does exist. For both classes of polynomials we prove an expected case bit complexity bound of $\tilde{\mathcal{O}}_B(r d^2 \tau)$, where r is the number of real roots. A close related bound was speculated in [18], based on experimental evidence.

Our bounds are tighter than those of the worst case by two factors. In the course of this analysis, STURM is shown to be output-sensitive, with complexity proportional to the number of real roots in the given interval, even in the worst case. A similar bound appeared in [15].

Besides polynomials in the monomial basis, polynomials in the Bernstein basis are important in many applications, e.g. CAGD and geometric modeling. They are of the form $\sum_{i=0}^d \alpha_i \binom{d}{i} x^i (1-x)^{d-i}$. For the random polynomials that we consider, α_i are standard normals i.i.d. random variables, that is Gaussians with zero mean and variance one. Such polynomials are also important in Brownian motion [21]. In [2], they examine random polynomial systems; they also estimate the expected number of real roots of a poly-

Algorithm 1: SUBDIVISIONSOLVER($A, \mathcal{J}_0$)

Input: Square-free $A \in \mathbb{Z}[x]$, $\mathcal{J}_0 = [0, B]$
Output: A list of isolating intervals for the real roots of A in $\mathcal{J}_0$

```

1  INITIALIZESM( $A, \mathcal{J}_0$ )
2   $L \leftarrow \emptyset$ ,  $Q \leftarrow \emptyset$ ,  $Q \leftarrow \text{PUSH}(Q, \{A, \mathcal{J}_0\})$ 
3  while  $Q \neq \emptyset$  do
4       $\{f, \mathcal{J}\} \leftarrow \text{POP}(Q)$ 
5       $V \leftarrow \text{COUNT}_{\text{SM}}(f, \mathcal{J})$ 
6      switch  $V$  do
7          case  $V = 0$  continue
8          case  $V = 1$   $L \leftarrow \text{ADD}(L, \mathcal{J})$ 
9          case  $V > 1$ 
10              $\{f_L, \mathcal{J}_L\}, \{f_R, \mathcal{J}_R\} \leftarrow \text{SPLIT}_{\text{SM}}(f, \mathcal{J})$ 
11              $Q \leftarrow \text{PUSH}(Q, \{f_L, \mathcal{J}_L\})$ ,  $Q \leftarrow \text{PUSH}(Q, \{f_R, \mathcal{J}_R\})$ 
12  RETURN  $L$ 

```

mial in the Bernstein basis as $\sqrt{d}$, when the variance is $\binom{d}{i}$. This left open the case, see also [21], of smaller variance, that is polynomial and not exponential in d .

Our second contribution is to examine random polynomials in the Bernstein basis of degree d , with i.i.d. coefficients with mean zero and “moderate” variance $\Theta(1/\sqrt{d/(i(d-i))})$, for $d > i > 0$. Indeed, we have $1 \geq \sqrt{d/(i(d-i))} \geq 2/\sqrt{\pi d}$. We prove that the expected number of real roots of these polynomials is $\sqrt{2d} \pm \mathcal{O}(1)$. We conclude with experimental results which corroborate our analysis, and shows that these polynomials behave like polynomial with variance 1. This is the first step towards bounding the expected complexity of solving polynomials in the Bernstein basis.

The rest of the paper is structured as follows. First we specify our notation. Sec. 2 and 3 applies our expected-case analysis to estimating the real root separation bound, and to estimating the complexity of STURM solver. Sec. 4 determines the expected number of real roots of random polynomial in the Bernstein basis and supports our bounds by experimental results. The paper concludes with a discussion of open questions.

Notation. $\mathcal{O}_B$ means bit complexity and the $\tilde{\mathcal{O}}_B$ -notation means that we are ignoring logarithmic factors. For $A = \sum_{i=1}^d \alpha_i X^i \in \mathbb{Z}[X]$, $\text{dg}(A)$ denotes its degree. $\mathcal{L}(A)$ denotes an upper bound on the bitsize of the coefficients of A (including a bit for the sign). For $a \in \mathbb{Q}$, $\mathcal{L}(a) \geq 1$ is the maximum bitsize of the numerator and the denominator. Δ is the separation bound of A , that is the smallest distance between two (real or complex, depending on the context) roots of A .

2. SUBDIVISION-BASED SOLVERS

In order to make the presentation self-contained, we present in some detail the general scheme of the subdivision-based solvers. The pseudo-code of a such a solver is found in Alg. 1. Our exposition follows closely [11].

The input is a square-free polynomial $A \in \mathbb{Z}[x]$ and an interval $\mathcal{J}_0$, that contains the real roots of A which we wish to isolate; usually it contains all the positive real roots of A . In what follows, except if explicitly stated otherwise, we consider only the roots (real and/or complex) of A with positive real part, since similar results could be obtained for roots with negative real part using the transformation $x \mapsto -x$. Our goal is to compute rational numbers between the real roots of A in $\mathcal{J}_0$.

The algorithm uses a stack Q that contains pairs of the

form $\{f, J\}$. The semantics are that we want to isolate the real roots of f contained in interval J . $\text{PUSH}(Q, \{f, J\})$ inserts the pair $\{f, J\}$ to the top of stack Q and $\text{POP}(Q)$ returns the pair at the top of the stack and deletes it from Q . $\text{ADD}(L, J)$ inserts J to the list L of the isolating intervals.

There are 3 sub-algorithms with index SM, which have different specializations with respect to the subdivision method applied, namely STURM, DESCARTES, or BERNSTEIN. Generally, $\text{INITIALIZATION}_{\text{SM}}$ does the necessary pre-processing, $\text{COUNT}_{\text{SM}}(f, J)$ returns the number (or an upper bound) of the real roots of f in J , and $\text{SPLIT}_{\text{SM}}(f, J)$ splits J to two equal subintervals and possibly modifies f .

The complexity of the algorithm depends on the number of times the while-loop (Line 3 of Alg. 1) is executed and on the cost of $\text{COUNT}_{\text{SM}}(f, J)$ and $\text{SPLIT}_{\text{SM}}(f, J)$. At every step, since we split the tested interval to two equal sub-intervals, we may assume that the bitsize of the endpoints is augmented by one bit. If we assume that the endpoints of J_0 have bitsize τ , then at step h , the bitsize of the endpoints of $J \subseteq J_0$ is $\tau + h$.

Let n be the number of roots with positive real part, and r the number of positive real roots, so $r \leq n \leq d$. Let the roots with positive real part, be $\alpha_j = \Re(\alpha_j) + i\Im(\alpha_j)$, where $1 \leq j \leq n$ and the index denotes an ordering on the real parts. Let Δ_i be the smallest distance between α_i and another root of A , and $s_i = \mathcal{L}(\Delta_i)$. Finally, let the separation bound, i.e. the smallest distance between two (possibly complex) roots of A be Δ and its bitsize be $s = \mathcal{L}(\Delta)$.

2.1 Upper root bound

Before applying a subdivision-based algorithm, we should compute a bound, B , on the (positive) roots. We will express this bound as a function of the bitsize of the separation bound and the degree of the polynomial. There are various bounds for the roots of a polynomial, e.g. [36, 16, 26], and references therein. For our analysis we use the following bound [16] on the positive real parts of the roots,

$$B = \left\lceil 2 \max_{\alpha_i < 0} \min_{\alpha_k > 0, k > i} \left| \frac{\alpha_i}{\alpha_k} \right|^{1/(k-i)} \right\rceil$$
, for which we have the estimation [16, 33] $\alpha_r \leq \Re(\alpha_n) < B < \frac{8d}{\ln 2} \Re(\alpha_n)$. The bound can be computed in $\tilde{O}_B(d^2 \tau)$.

If we multiply the polynomial by x , then 0 is a root. By definition of s , we have $|\log(|\Re(\alpha_i) - \Re(\alpha_j)|)| \leq s$, for any $i \neq j$. Hence, we have the following inequalities

$$\begin{array}{rclcl} \Re(\alpha_1) & - & 0 & \leq & 2^s \\ \Re(\alpha_2) & - & \Re(\alpha_1) & \leq & 2^s \\ & & \vdots & & \\ \Re(\alpha_{n-1}) & - & \Re(\alpha_{n-2}) & \leq & 2^s \\ \Re(\alpha_n) & - & \Re(\alpha_{n-1}) & \leq & 2^s \quad (+) \\ \hline & & \Re(\alpha_n) & \leq & n 2^s \end{array}$$

Thus, we have $B < \frac{8d}{\ln 2} \Re(\alpha_n) < \frac{8d}{\ln 2} n 2^s < 16 d^2 2^s < d^2 2^{4+s}$. Hence, we can deduce that $\mathcal{L}(B) = \mathcal{O}(s + \lg d)$.

Lemma 2.1. *Let $A \in \mathbb{Z}[x]$, where $\text{dg}(A) = d$ and $\mathcal{L}(A) = \tau$. We can compute a bound, B , on the positive real parts of the roots of A , for which it holds $B < d^2 2^{4+s}$, and $\mathcal{L}(B) = \mathcal{O}(s + \lg d)$.*

Remark 2.2. *In the worst case, the asymptotics of, more or less, all root bounds in the literature, e.g. [36, 16, 26], are same, since $B \leq \max_i |\alpha_i| \leq 2^\tau$, and $\mathcal{L}(B) \leq \tau$. However, it is very important in practice to have good initial bounds. Good initial estimations of the roots can speed up the implementation by 20% [22].*

3. ON EXPECTED COMPLEXITY

Expected complexity aims to capture and quantify the property for an algorithm to be fast for most inputs and slow for some rare instances of these inputs. Let E denote the set of inputs, and assume it is equipped with a probability measure μ ; then let $c(I)$ denote the usual worst-case complexity of the considered algorithm for input I . By definition, the expected complexity is the integral $\int_E c(I) \mu(I)$.

In our setting the set E depends on a parameter d (the degree of the input polynomial), and we are interested in the asymptotic expected complexity when d tends to infinity. Each E_d is equipped with a probability measure μ_d (also called distribution) of the sequence of the (normalized) coefficients of the input polynomial and we consider the cases where there exists a limit distribution.

3.1 Strategy and Independence

A natural strategy is to decompose E_d into two subsets G_d and R_d (G stands for generic and R for rare), such that $c(I)$ is small for $I \in G_d$ while $\mu_d(I)$ is very small for $I \in R_d$ and moreover the two partial integrals $\int_{G_d} c(I) \mu_d(I)$ and $\int_{R_d} c(I) \mu_d(I)$ are balanced or at least both small.

We face another difficulty. Classical properties and estimates in Probability theory are often expressed for a sequence of independent variables (i.i.d.) but most natural bijective transformations performed in Computer Algebra do not respect independence. For instance, if X and Y are independent random variables, then $U := X + Y$ and $V := X - Y$ are not independent. In our setting, even if we consider a model of distribution of coefficients which assumes that they are i.i.d., then this does not imply that the roots are i.i.d. and we cannot apply usual tools or estimates. However, as we are interested in asymptotic behavior, for some models of distribution of coefficients it happens that the limit distribution of the roots behave almost like a set of independent variables, i.e. they have very weak correlation. So we can invoke general classical estimates for our analysis.

When this is not the case, a useful tool is the two-point, or multi-point, correlation function. They express the defect of independence between a set of random variables and classically serve, e.g., to compute standard deviations.

Hereafter, we restrict ourselves to models of distribution of coefficients, hence induced distribution of roots, for which the corresponding probability measures and correlation functions have already been studied. Hopefully these models will provide good approximations for the situations encountered in the many applications.

3.2 SO(2) polynomials

We consider the univariate polynomial $A = \sum_{i=0}^d a_i x^i$, the coefficients of which are i.i.d. normals with mean zero and variances $\binom{d}{i}$, where $0 \leq i \leq d$. Alternatively, we could consider A as $A = \sum_{i=0}^d \sqrt{\binom{d}{i}} a_i x^i$, where a_i are i.i.d. standard normals. These polynomials are considered by Edelman and Kostlan [9] to be “the more natural definition of a random polynomial”. They are called SO(2) because the joint probability distribution of their zeros is SO(2) invariant, after homogenization. In [31] they are called *binomial*. Let $\rho(t) = \frac{\sqrt{d}}{\pi(1+t^2)}$ be the true density function, i.e. the expected number of real zeros per unit length at a point $t \in \mathbb{R}$. The expected number r of real roots of A is given by $r = \int_{\mathbb{R}} \rho(t) dt = \sqrt{d}$ [9]. Let α_j be the real roots of A in

their natural ordering, where $1 \leq j \leq r$.

We define the straightened zeros of A as

$$\zeta_j = \mathcal{P}(\alpha_j) = \sqrt{d} \arctan(\alpha_j)/\pi, j = 1, \dots, r,$$

in bijective correspondence with the real roots α_j of the random polynomial, where $\mathcal{P}(t) = \int_0^t \rho(u) du$. Moreover, the ordering is preserved. The straightened zeros are uniformly distributed on the circle of length $2\sqrt{d}$ [4, sec.5]. This is a strong property and implies that the joint probability distribution density function of two, resp. m , (distinct) straightened zeros coincides with their 2-point, resp. m -point, correlation function [4].

Proposition 3.1. [4, Thm. 5.1] *Following the previous notation, as $d \rightarrow \infty$ the limit 2-point correlation of the straightened zeros is $k(s_1, s_2) \rightarrow \pi^2 |s_1 - s_2|/4$, when $s_1 - s_2 \rightarrow 0$.*

Let $\Delta(\alpha) = \min_{1 \leq i < r} \{\alpha_{i+1} - \alpha_i\}$ and $\Delta(\zeta) = \min_{1 \leq i < r} \{\zeta_{i+1} - \zeta_i\}$ be the separation bound of the real roots of A and the straightened zeros, respectively. We consider each straightened zero uniformly distributed on a straight-line interval of length $2\sqrt{d}$. For two such zeros, we can consider one horizontal and one vertical such interval, defining a square, which represents their joint probability space. Since the real roots are naturally ordered, if two of them lie in a given infinitesimal interval, they must be consecutive.

Let Z be a zone bounded above and below by a diagonal at vertical distance l from the main diagonal of the unit square. The probability $\Pr[\Delta(\zeta) \leq l]$ that there exist two zeros lying in a given interval of infinitesimal length l tends to the integral of $k(s_1, s_2)$ over the straightened zeros lying in Z , as $d \rightarrow \infty$:

$$\begin{aligned} \Pr[\Delta(\zeta) \leq l] &\rightarrow \int_Z k(s_1, s_2) ds_1 ds_2 \\ &= 2 \int_0^{2\sqrt{d}} ds_1 \int_{s_1}^{s_1+1} k(s_1, s_2) ds_2 \\ &= \frac{\pi^2}{2} \int_0^{2\sqrt{d}} ds_1 \int_{s_1}^{s_1+1} |s_1 - s_2| ds_2 = \frac{\pi^2 \sqrt{d}}{2} l^2, \end{aligned}$$

where the first integral is over all straightened zeros, which lie in an interval of size $2\sqrt{d}$. Notice that $k(s_1, s_2)$ is essentially the joint probability density function of two real roots. Using Markov's inequality, e.g. [28] we have $\Pr[\Delta(\zeta) \geq l] \leq E[\Delta(\zeta)]/l$, so

$$E[\Delta(\zeta)] \geq l \Pr[\Delta(\zeta) \geq l] = l - l \Pr[\Delta(\zeta) < l] > l - \frac{\pi^2 \sqrt{d}}{2} l^3.$$

This bounds the asymptotic expected separation conditioned on the hypothesis that it tends to zero, as $d \rightarrow \infty$. If we choose $l = 1/(d^c \tau)$, where $c \geq 1$ is a (small) constant, which is in accordance with the assumption of $l \rightarrow 0$, then $E[\Delta(\zeta)] > \frac{1}{d^c \tau} - \frac{\pi^2}{2 d^{3c-1/2} \tau^3}$.

$$\begin{aligned} E[\Delta(\zeta)] &= E[\min_{1 \leq i < r} \{\zeta_{i+1} - \zeta_i\}] = \\ &= \frac{\sqrt{d}}{\pi} E[\min_{1 \leq i < r} \{\arctan(\alpha_i) - \arctan(\alpha_{i+1})\}] = \\ &= \frac{\sqrt{d}}{\pi} E[\min_{1 \leq i < r} \{\arctan\left(\frac{\alpha_i - \alpha_{i+1}}{1 + \alpha_i \alpha_{i+1}}\right)\}] > \frac{1}{d^c \tau} - \frac{\pi^2}{2 d^{3c-1/2} \tau^3} \Leftrightarrow \\ &E[\min_{1 \leq i < r} \{\arctan\left(\frac{\alpha_i - \alpha_{i+1}}{1 + \alpha_i \alpha_{i+1}}\right)\}] > \frac{\pi}{d^{c+1/2} \tau} - \frac{\pi^3}{2 d^{3c} \tau^3}. \end{aligned}$$

Function $\arctan$ is strongly monotone, and $1 + \alpha_i \alpha_{i+1} \geq 1$, for all i , except where α_i is the largest negative root and α_{i+1} is the smallest positive root. But we can treat this case

separately, since zero is an obvious separation point.

$$\begin{aligned} E[\min_{1 \leq i < r} \{\alpha_i - \alpha_{i+1}\}] &\geq E[\min_{1 \leq i < r} \left\{ \frac{\alpha_i - \alpha_{i+1}}{1 + \alpha_i \alpha_{i+1}} \right\}] > \\ &> \tan\left(\frac{\pi}{d^{c+1/2} \tau} - \frac{\pi^3}{2 d^{3c} \tau^3}\right) \geq \frac{\pi}{d^{c+1/2} \tau} - \frac{\pi^3}{2 d^{3c} \tau^3}, \end{aligned}$$

where the latter inequality follows from the series expansion $\tan x = x + x^3/3 + \dots$ for $x \in (0, \pi/2)$.

Lemma 3.2. *Let $A \in \mathbb{Z}[x]$ of degree d , the coefficients of which are i.i.d. variables that follow a normal distribution with variances $\binom{d}{i}$, then for the expected value of the separation bound of the real roots it holds $E[\Delta] > \frac{\pi}{d^{c+1/2} \tau} - \frac{\pi^3}{2 d^{3c} \tau^3}$, for a constant $c \geq 1$, and $E[s] = E[\mathcal{L}(\Delta)] = \mathcal{O}(\lg d + \lg \tau)$.*

3.3 Weyl polynomials

We consider random polynomials, known as Weyl polynomials, which are of the form

$$A = \sum_{i=0}^d a_i x^i / \sqrt{i!},$$

where the coefficients a_i are independent standard normals. Alternatively, we could consider A as $A = \sum_{i=0}^d a_i x^i$, where a_i are normals of mean zero and variance $1/\sqrt{i!}$. The density of the real roots of Weyl polynomials is

$$\rho(t) = \frac{1}{\pi} \sqrt{1 + \frac{t^{2d}(t^2 - d - 1)}{e^{t^2} \Gamma(n+1, t^2)} - \frac{t^{4d+2}}{(e^{t^2} \Gamma(n+1, t^2))^2}},$$

where Γ is the incomplete gamma function. The expected number of real roots is $r = \int_{\mathbb{R}} \rho(t) dt \sim \frac{2}{\pi} \sqrt{d}$ [31], where the higher order terms of the number of real roots are not explicitly known up to now.

The asymptotic density, for $d \rightarrow \infty$, is

$$\rho(t) = \begin{cases} \pi^{-1}, & |t| \ll \sqrt{d} \\ \frac{d}{\pi t^2}, & |t| \gg \sqrt{d} \end{cases} \quad (1)$$

A useful observation is that the density of the real roots of the Weyl polynomials is similar to the density of the real eigenvalues of Ginibre random matrices, that is $d \times d$ matrices with elements Gaussian i.i.d. random variables [9, 31].

We consider only the real zeros of A that are inside the disc centered at the origin with radius $\sqrt{d}$ since outside the disc there is only a constant number of them. In this case the density is represented by the first branch of (1).

We work as in the case of the $SO(2)$ polynomials. Now $\mathcal{P}(t) = \int_0^t \rho(u) du = t/\pi$. The straightened zeros, ζ_i , are given by

$$\zeta_i = \mathcal{P}(\alpha_i) = \alpha_i/\pi,$$

and they are uniformly distributed in $[0, \sqrt{d}/\pi]$ [31]. The joint probability distribution density function of two straightened zeros coincides with their 2-point correlation function.

Proposition 3.3. [31] *Under the previous notation, as $d \rightarrow \infty$ the limit 2-point correlation of the straightened zeros is $w(s_1, s_2) \rightarrow |s_1 - s_2|/(4\pi)$, when $s_1 - s_2 \rightarrow 0$.*

Working as in the case of the $SO(2)$ polynomials, the probability $\Pr[\Delta(\zeta) \leq l]$ that there exist two roots lying in a given interval of infinitesimal length l tends to the integral

of $w(s_1, s_2)$ over the straightened zeros lying in Z , as $d \rightarrow \infty$:

$$\begin{aligned} \Pr[\Delta(\zeta) \leq l] &= \int_Z w(s_1, s_2) ds_1 ds_2 \\ &= \int_0^{\sqrt{d}/\pi} \int_{s_1-1}^{s_1+1} w(s_1, s_2) ds_1 ds_2 = \frac{l^2 \sqrt{d}}{4\pi^2}, \end{aligned}$$

and using Markov's inequality

$$\Pr[\Delta(\zeta) \geq l] \leq E[\Delta(\zeta)]/l \iff E[\Delta] > l - \frac{\sqrt{d}}{4\pi^2} l^3.$$

If we choose $l = 1/(d^c \tau)$, where $c \geq 1$ is a (small) constant, we get $E[\Delta(\zeta)] > \frac{1}{d^c \tau} - \frac{1}{4\pi^2 d^{3c-1/2} \tau^3}$ and $E[\Delta(\alpha)] > \frac{\pi}{d^c \tau} - \frac{1}{4\pi d^{3c-1/2} \tau^3}$.

Lemma 3.4. *Let $A \in \mathbb{Z}[x]$ of degree d , the coefficients of which are i.i.d. variables that follow a normal distribution with variances $1/i!$, then for the expected value of the separation bound of the real roots it holds $E[\Delta] > \frac{\pi}{d^c \tau} - \frac{1}{4\pi d^{3c-1/2} \tau^3}$ and $E[s] = E[\mathcal{L}(A)] = \mathcal{O}(\lg d + \lg \tau)$.*

3.4 The STURM solver

Probably the first certified subdivision-based algorithm is the algorithm by Sturm, circa 1835, based on his theorem: In order to count the number of real roots of a polynomial in an interval, one evaluates a negative polynomial remainder sequence of the polynomial and its derivative over the left endpoint of the interval and counts the number of sign variations. We do the same for the right endpoint; the difference of sign variations is the number of real roots.

We assume that the positive real roots are contained in $[0, B]$ (Sec. 2.1). If there are r of them, then we need to compute $r-1$ separating points. The magnitude of the separation points is at most $\frac{1}{2} \Delta_j$, for $1 \leq j \leq r$, and to compute each we need $\lceil \lg \frac{2B}{\Delta_j} \rceil$ subdivisions, performing binary search in the initial interval. Let T be the binary tree that corresponds to the execution of the algorithm and $\#(T)$ be the number of its nodes, or in other words the total number of subdivisions:

$$\#(T) = \sum_{j=1}^r \left\lceil \lg \frac{2B}{\Delta_j} \right\rceil \leq 2r + r \lg B - \sum_{j=1}^r \lg \Delta_j. \quad (2)$$

Using Lem. 2.1, we deduce that $\#(T) = \mathcal{O}(rs + r \lg(d))$.

The Sturm sequence should be evaluated over a rational number, the bitsize of which is at most the bitsize of the separation bound. Using fast algorithms [23, 29] this cost is $\tilde{\mathcal{O}}_B(d^2(\tau + s))$; to derive the overall complexity we should multiply it by $\#(T)$. Notice that for the evaluation we use the sequence of the quotients, which we computed in $\tilde{\mathcal{O}}_B(d^2 \tau)$ [23, 29], and not the whole Sturm sequence, which can be computed in $\tilde{\mathcal{O}}_B(d^3 \tau)$, e.g. [7].

The previous discussion allows us to express the bit complexity of STURM not only as a function of the degree and the bitsize, but also using the number of real roots and the (logarithm of) separation bound. This complexity is output sensitive, and is of independent interest, although it leads to a loose worst-case bound.

Lemma 3.5. *Let $A \in \mathbb{Z}[x]$, $\text{dg}(A) = d$, $\mathcal{L}(A) = \tau$ and let s be the bitsize of its separation bound. Using STURM, we isolate the real roots of A with worst-case complexity $\tilde{\mathcal{O}}_B(r d^2(s^2 + \tau s))$, where r is the number of real roots.*

In the worst case $s = \mathcal{O}(d\tau)$, and to derive the worst case complexity bound for STURM, $\tilde{\mathcal{O}}_B(d^4 \tau^2)$, we should also take into account that $ds = \mathcal{O}(d\tau)$.

To derive the expected complexity we should consider two cases for the separation bound, that is, smaller or bigger than $l = 1/(d^c \tau)$, where $c \geq 1$ is a small constant that shall be specified later.

In the first case, that is $\Delta \leq l = 1/(d^c \tau)$, the real roots are not well separated, so we rely on the worst case bound for isolating them, that is $\tilde{\mathcal{O}}_B(d^4 \tau^2)$. This occurs with probability $\Pr[\Delta \leq l] = \Theta(\sqrt{d} l^2) = \Theta(\frac{1}{d^{2c-1/2} \tau^2})$, by the computations of Sec. 3.2 and Sec.3.3. This probability is very small.

For the second case, since $\Delta > 1/(d^c \tau)$ we deduce $s = \mathcal{O}(\lg d + \lg \tau)$. The complexity of isolating the real roots, following Lem. 3.5 is $\tilde{\mathcal{O}}_B(r d^2 \tau)$. The computations in Sec. 3.2 and Sec.3.3 suggest that this case occurs with probability $\Pr[\Delta > l] = 1 - \Theta(\sqrt{d} l^2) = 1 - \Theta(\frac{1}{d^{2c-1/2} \tau^2})$, which is close to one.

The expected-case complexity bound of STURM is

$$\tilde{\mathcal{O}}_B \left(\left(1 - \frac{1}{d^{2c-1/2} \tau^2}\right) \cdot r d^2 \tau + \frac{1}{d^{2c-1/2} \tau^2} \cdot d^4 \tau^2 \right) = \tilde{\mathcal{O}}_B(r d^2 \tau),$$

for any $c \geq 1$, by using $\sqrt{d} = \tilde{\mathcal{O}}(r\tau)$, which follows from the expected number of real roots. To avoid using this expected number, it suffices to set $c \geq 2$.

Theorem 3.6. *Let $A \in \mathbb{Z}[x]$, where $\text{dg}(A) = d$, $\mathcal{L}(A) = \tau$. If A is either a $\text{SO}(2)$ or a Weyl random polynomial, then the expected complexity of STURM solver is $\tilde{\mathcal{O}}_B(r d^2 \tau)$.*

In practice, the Sturm sequence is used and not the quotient sequence. The cost of the former is $\tilde{\mathcal{O}}_B(d^3 \tau)$ which dominates the bound of Th. 3.6. This explains the empirical observations that most of the execution time of STURM solver is spend on the construction of the Sturm sequence.

4. RANDOM BERNSTEIN POLYNOMIALS

We compute the expected number of real roots of polynomials with random coefficients, represented in the Bernstein basis. We start with some lemmata.

Lemma 4.1. *For $k \leq n$, non-negative integers, it holds*

$$\sum_{j=0}^n \binom{kn}{kj} x^{kj} = \frac{1}{k} \sum_{j=0}^{k-1} (x + e^{i \frac{2\pi j}{k}})^{kn}.$$

Proof: We consider the RHS of the equality. For a specific j we expand the summand, and get terms of the form

$$\binom{kn}{\mu} x^{kn-\mu} e^{i \frac{2\pi j}{k} \mu}, \quad 0 \leq \mu \leq kn.$$

There are $kn + 1$ such terms. Recall that $e^{i 2\pi} = 1$. Let $\mu = \lambda k + \nu$, where $1 \leq \nu \leq k-1$, $0 \leq \lambda < n$, then

$$\begin{aligned} \binom{nk}{\lambda k + \nu} x^{kn-\lambda k - \nu} e^{i \frac{2\pi j}{k} (\lambda k + \nu)} &= \\ \binom{nk}{\lambda k + \nu} x^{kn-\lambda k - \nu} e^{i \frac{2\pi j}{k} \lambda k} e^{i \frac{2\pi j}{k} \nu} &= \binom{nk}{\lambda k + \nu} x^{kn-\lambda k - \nu} e^{i \frac{2\pi j}{k} \nu}. \end{aligned}$$

If we sum all these terms over j , we get

$$\sum_{j=0}^{k-1} \binom{nk}{\lambda k + \nu} x^{kn-\lambda k - \nu} e^{i \frac{2\pi j}{k} \nu} = \binom{nk}{\lambda k + \nu} x^{kn-\lambda k - \nu} \sum_{j=0}^{k-1} e^{i \frac{2\pi j}{k} \nu} = 0,$$

since $\sum_{j=0}^{k-1} e^{i \frac{2\pi j}{k} \nu} = 0$.

Let $\mu = \lambda k$. In this case, we have

$$\binom{nk}{\lambda k} x^{kn-\lambda k} e^{i \frac{2\pi j}{k} \lambda k} = \binom{nk}{\lambda k} x^{kn-\lambda k} = \binom{nk}{k(n-\lambda)} x^{k(n-\lambda)}$$

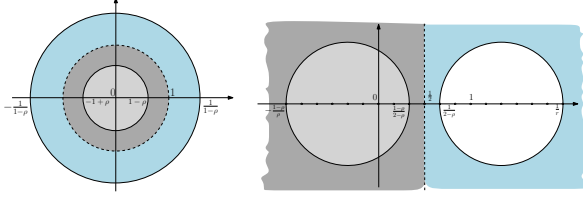


Figure 1. The transformation $z := \frac{y}{y+1}$ in $\mathbb{C}$.

Notice that $0 \leq \lambda \leq n$. Summing up over all λ and all j , and multiplying by $1/k$ we get the LHS. $\square$

Lemma 4.2. For non-negative integers n, k, p ,

$$\frac{\binom{n}{k}^p}{\binom{pn}{pk}} \approx \sqrt{p} \left(\frac{n}{2\pi} \right)^{p-1} \sqrt{\left(\frac{1}{k(n-k)} \right)^{p-1}}.$$

Proof: The proof follows easily from Stirling's approximation $n! \approx \sqrt{2\pi n} \left(\frac{n}{e} \right)^n$. $\square$

More accurate results could be obtained if the more precise expression $\sqrt{2\pi n} \left(\frac{n}{e} \right)^n e^{\frac{1}{12n+1}} < n! < \sqrt{2\pi n} \left(\frac{n}{e} \right)^n e^{\frac{1}{12n}}$, is considered.

4.1 The expected number of real roots

We aim to count the real positive roots of a random polynomial in the Bernstein basis of degree d , i.e.

$$\hat{P} := \sum_{k=0}^d b_k \binom{d}{k} z^k (1-z)^{d-k}, \quad (3)$$

where we assume that $\hat{P}(0)\hat{P}(1) \neq 0$, and $\{b_k\}$ is an array of random real numbers, following the normal distribution, with “moderate” standard deviation, which shall be specified below.

We introduce a suitable change of coordinates, $z := y/(y+1)$, to transform a polynomial in the Bernstein basis into one in the monomial basis, by setting $P = (1+y)^d \hat{P}(y)$. Now, P and $\hat{P}$ have the same number of real roots, and

$$P = \sum_{k=0}^d b_k \binom{d}{k} y^k.$$

Even though the number of real roots does not change, their distribution over the real axis does, see Fig. 1. In particular, we can now apply the techniques already used by Edelman, Kostlan, and others for counting the number (and, eventually, the limit distribution) of real roots. Of course, by symmetry, the expected number of positive and negative real roots is equal.

By Lem. 4.2, setting $p = 2$ and $n = d$ we deduce:

$$\binom{d}{k} \approx \sqrt{\frac{d}{\pi}} \sqrt{\frac{1}{k(d-k)}} \sqrt{\binom{2d}{2k}} =: \sqrt{S_k} \sqrt{\binom{2d}{2k}}. \quad (4)$$

It holds that $\sqrt{2/\sqrt{\pi d}} \leq \binom{d}{k}/\sqrt{\binom{2d}{2k}} = \sqrt{S_k} \leq 1$. To prove this, notice that S_k is decreasing from 1 to $d/2$ and increasing from $d/2$ to $d-1$. Hence the lower bound is attained at $k = d/2$ and the upper bound at $k = 1$ and $k = d-1$.

Since S_k is small compared to $\binom{d}{k}$, it is reasonable to assume that omitting it will make only a negligible change in the asymptotic analysis.

Let $y = x^2$, with $x > 0$. Now the problem at hand to count the positive real roots of

$$P = \sum_{k=0}^d a_k \sqrt{\binom{2d}{2k}} x^{2k}.$$

We need the following proposition

Proposition 4.3. [9] Let $v(t) = (f_0(t), \dots, f_n(t))^T$ be a vector of differentiable functions and $c_0, \dots, c_n$ elements of a multivariate normal distribution with zero mean and covariance matrix C . The expected number of real zeros on an interval (or a measurable set) I of the equation $c_0 f_0(t) + \dots + c_n f_n(t) = 0$, is

$$\int_I \frac{1}{\pi} \|\mathbf{w}'(t)\| dt, \quad \mathbf{w} = \mathbf{w}(t)/\|\mathbf{w}(t)\|.$$

where $\mathbf{w}(t) = C^{1/2}v(t)$. In logarithmic derivative notation, this is

$$\frac{1}{\pi} \int_I \sqrt{\frac{\partial^2}{\partial x \partial y} \log(v(x)^T C v(x))}|_{x=y=t} dt.$$

For computing the integral in Prop. 4.3, we shall use the logarithmic derivative notation. Following Prop. 4.3, $f_{2i}(t) = \sqrt{\binom{2d}{2i}} x^{2i}$ and $f_{2i+1}(t) = 0$, $c_{2i} = a_i$ and $c_{2i+1} = 0$, where $0 \leq i \leq d$, and the variance is 1. Then,

$$v(x)^T C v(y) = \sum_{k=0}^d \binom{2d}{2k} (xy)^{2k}.$$

We consider function

$$f(z) = \sum_{k=0}^d \binom{2d}{2k} z^{2k}.$$

By Lem. 4.1, for $k = 2$, we have $f(z) = \frac{1}{2}((1+z)^{2d} + (1-z)^{2d})$ and so $f'(z) = d(z+1)^{2d-1} + d(z-1)^{2d-1}$, $f''(z) = d(2d-1)(z+1)^{2d-2} + d(2d-1)(z-1)^{2d-2}$.

The following quantities are also relevant $ff' = \frac{1}{2}d(z+1)^{4d-1} + dz(z^2-1)^{2d-1} + \frac{1}{2}d(z-1)^{4d-1}$, $ff'' = \frac{1}{2}d(2d-1)(z+1)^{4d-2} + d(2d-1)(z^2+1)(z^2-1)^{2d-2} + \frac{1}{2}d(2d-1)(z-1)^{4d-2}$, and $(f')^2 = d^2(z+1)^{4d-2} + 2d^2(z^2-1)^{2d-1} + d^2(z-1)^{4d-2}$.

It holds that

$$\partial_x \partial_y (\log f(x, y)) = \frac{f' f + xy f'' f - xy (f')^2}{f^2} = \frac{A}{f^2},$$

with

$$\begin{aligned} A &= d(z+1)^{4d-2} \left(\frac{1}{2}(z+1) + \frac{1}{2}(2d-1)z - zd \right) \\ &\quad + d(z^2-1)^{2d-2} (z(z^2-1) + (2d-1)z(z^2+1) - 2d(z^2-1)z) \\ &\quad - d(z-1)^{4d-2} \left(\frac{1}{2}(z-1) + \frac{1}{2}(2d-1)z - zd \right) \\ &= \frac{1}{2}d \left((z+1)^{4d-2} + 4(2d-1)z(z^2-1)^{2d-2} - (z-1)^{4d-2} \right). \end{aligned}$$

If we let $z = t^2$, then

$$\begin{aligned} \frac{A(t^2)}{f(t^2)^2} &= \frac{\frac{1}{2}d((1+t^2)^{4d-2} + 4(2d-1)t^2(t^4-1)^{2d-2} - (t^2-1)^{4d-2})}{\frac{1}{4}((1+t^2)^{2d} + (1-t^2)^{2d})^2} \\ &= 2d \frac{1}{(1+t^2)^2} \frac{1+(2d-1)\left(\frac{2t^2}{1+t^2}\right)^2 \left(\frac{1-t^2}{1+t^2}\right)^{2d-2} - \left(\frac{1-t^2}{1+t^2}\right)^{4d-2}}{\left(1+\left(\frac{1-t^2}{1+t^2}\right)^2\right)^2}. \end{aligned}$$

We consider the substitutions $t = \tan \frac{\theta}{2}$, $\tan \theta = \frac{2t}{1-t^2}$, $\sin \theta = \frac{2t}{1+t^2}$, $\cos \theta = \frac{1-t^2}{1+t^2}$, and $\frac{d\theta}{dt} = \frac{dt}{1+t^2}$. Then

$$\frac{A}{f(t^2)^2} = 2d \frac{1}{(1+t^2)^2} \frac{1+(2d-1)\sin^2 \theta (\cos \theta)^{2d-2} - (\cos \theta)^{4d-2}}{(1+(\cos \theta)^2)^2}.$$

The expected number of positive real roots is given by

$$\begin{aligned} I &= \frac{1}{\pi} \int_0^\pi \frac{\sqrt{A}}{f(t^2)} dt \\ &= \frac{1}{\pi} \int_0^\pi \sqrt{2d} \frac{\sqrt{1+(2d-1)\sin^2\theta(\cos\theta)^{2d-2}-(\cos\theta)^{4d-2}}}{1+(\cos\theta)^{2d}} d\theta. \end{aligned}$$

Performing the change $\theta \mapsto \pi - \theta$, we notice that I equals twice the integral between 0 and $\pi/2$. Hence, the expected number of positive real roots of P in $(0, 1)$ equals that in $(1, \infty)$. Hence,

$$I = \frac{\sqrt{2d}}{\pi} \int_0^{\pi/2} \frac{\sqrt{1+(2d-1)\sin^2\theta(\cos\theta)^{2d-2}-(\cos\theta)^{4d-2}}}{1+(\cos\theta)^{2d}} d\theta.$$

Now we will bound the integral as $d \rightarrow \infty$. Applying the triangular inequality and noticing that $1 + (\cos\theta)^{4d-2} \leq 1$ and $a + \cos\theta^{2d} \geq 1$, we get

$$\begin{aligned} I &\leq \frac{\sqrt{2d}}{\pi} \left[\int_0^{\pi/2} \sqrt{1} d\theta + \int_0^{\pi/2} \sqrt{2d-1} \sin\theta (\cos\theta)^{d-1} d\theta \right] \\ &= \frac{\sqrt{2d}}{\pi} \left(\frac{\pi}{2} + \sqrt{2d-1} \frac{1}{d} \right) = \frac{\sqrt{2d}}{2} \left(1 + \frac{\sqrt{2d-1}}{d} \right) \\ &\leq \frac{\sqrt{2d}}{2} \left(1 + \frac{1}{\pi} \sqrt{\frac{2}{d}} \right) \leq \frac{\sqrt{2d}}{2} + \frac{1}{\pi}. \end{aligned}$$

For a lower bound, we neglect the positive term $(2d-1)\sin^2\theta(\cos\theta)^{2d-2}$, and notice that $\sqrt{1+(\cos\theta)^{4d-2}} \geq 1 + (\cos\theta)^{2d-1} \geq 1 + (\cos\theta)^{2d-2} = (1+(\cos\theta)^{d-1})(1-(\cos\theta)^{d-1})$, and $\frac{1+(\cos\theta)^{d-1}}{1+(\cos\theta)^{2d}} \geq 1$.

Lemma 4.4.

$$W(n) := \int_0^{\pi/2} (\cos\theta)^n d\theta \leq \frac{2}{\sqrt{\pi}} \frac{1}{\sqrt{d+1}}.$$

Proof: We need the following inequality [6] on Wallis' cosine formula:

$$\frac{1}{\sqrt{\pi(k+4\pi^{-1}-1)}} \leq \frac{1 \cdot 3 \cdot 5 \cdots (2k-1)}{2 \cdot 4 \cdot 6 \cdots (2k)} \leq \frac{1}{\sqrt{\pi(k+1/4)}}.$$

If n is even then $W(n) = \frac{\pi}{2} \frac{(n-1)!!}{n!} = \frac{\pi}{2} \frac{1 \cdot 3 \cdot 5 \cdots (n-1)}{2 \cdot 4 \cdot 6 \cdots (n)} \leq \frac{\pi}{\sqrt{\pi(2n+1)}} \leq \frac{2}{\sqrt{\pi}} \frac{1}{\sqrt{d+1}}.$

If n is odd then $W(n) = \frac{(n-1)!!}{n!} = \frac{2 \cdot 4 \cdot 6 \cdots (n-3)(n-1)}{1 \cdot 3 \cdot 5 \cdots (n-2)n} \leq \sqrt{\pi(k+4\pi^{-1}-1)} \cdot \frac{1}{n} \leq \frac{2}{\sqrt{\pi}} \frac{1}{\sqrt{d+1}}.$ $\square$

Using the lemma, $\int_0^{\pi/2} (\cos\theta)^{d-1} d\theta \leq \frac{2}{\sqrt{\pi}} \frac{1}{\sqrt{d}}$, so:

$$\begin{aligned} I &\geq \frac{\sqrt{2d}}{\pi} \int_0^{\pi/2} 1 - (\cos\theta)^{d-1} d\theta \\ &\geq \frac{\sqrt{2d}}{2} \left(1 - \frac{4}{\pi\sqrt{\pi}} \frac{1}{\sqrt{d}} \right) \geq \frac{\sqrt{2d}}{2} - \sqrt{\frac{8}{\pi^3}}. \end{aligned}$$

Hence $I = \frac{\sqrt{2d}}{2} \pm \mathcal{O}(1)$ and we can state the following:

Theorem 4.5. *The expected number of real roots of a random polynomial $P = \sum_{k=0}^d a_k \sqrt{\binom{2d}{2k}} x^{2k}$, where a_k are standard normals i.i.d. random variables, is $\sqrt{2d} \pm \mathcal{O}(1)$.*

By employing (4) and considering $\sqrt{S_k}$ as part of the deviation, we have the following:

Corollary 4.6. *The expected number of real roots of a random polynomial in the Bernstein basis, Eq. (3), the coefficients of which are normal i.i.d. random variables with mean 0 and variance $1/S_k = 1/\sqrt{\pi k(d-k)}$, is $\sqrt{2d} \pm \mathcal{O}(1)$.*

In Table 1 we present the results of experiments with polynomials in the Bernstein basis (see Eq. (3)), of degree ≤ 1000 , the coefficients of which are i.i.d. random variables following the standard normal distribution, that is mean zero and

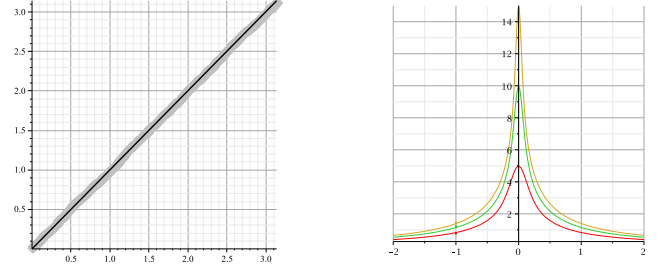


Figure 2. Left: Function $\arccos(2t-1)$ of real roots in $(0, 1)$, against uniform distribution in $(0, \pi)$. Right: Density of polynomials in the Bernstein basis for $d \in \{5, 10, 15\}$.

variance 1. For each degree we tested 100 polynomials. The first column is the degree, while the second is the expected number of real roots predicted by Cor. 4.6 which assumes variance $1/S_k$. The third column is the average number of real roots computed. Our experiments support the following conjecture:

Conjecture 4.7. *The expected number of real roots of a random polynomial in the Bernstein basis, Eq. (3), the coefficients of which are standard normal i.i.d. random variables, that is with mean 0 and variance 1, is $\sqrt{2d} \pm \mathcal{O}(1)$.*

Columns 4-7 of Tab. 1 corresponds to the average number of real roots in the intervals $(-\infty, -1)$, $(-1, 0)$, $(0, 1)$ and $(1, \infty)$, respectively. For these experiments we took random polynomials in the monomial basis and converted them to the Bernstein basis. The roots of a random polynomial in the monomial basis, under the assumptions of [17], concentrate around the unit circle. The symmetry of the density suggests that each of the intervals $(-1/(1-\rho), -1)$, $(-1, -1+\rho)$, $(1-\rho, 1)$, and $(1, 1/(1-\rho))$, contains on the average $1/4$ of the real roots (Fig. 1, left). If we apply the transformation $x \mapsto x/(x+1)$ (Fig. 1, right) to transform the polynomial to the Bernstein basis, then $3/4$ of the real roots are positive, $1/2$ of them are in $(0, 1)$ and $1/4$ in $(1, \infty)$. We refer to the last columns of Tab. 1 for experimental evidences of this.

As far as the distribution of the real roots in $(0, 1)$ is concerned, if we denote them by t_i , then $\arccos(2t_i - 1)$, behaves as the uniform distribution in $(0, \pi)$. In Fig. 2, we present the probability-probability plot, (using the `ProbabilityPlot` command of MAPLE) of this function of real roots of random polynomials in Bernstein basis, of degree 1000 (light grey line), against the theoretical uniform distribution (black line) in $(0, \pi)$. We observe that the lines almost match. For reasons of space, we postpone the discussion about the distribution of the roots.

5. CONCLUSIONS AND FUTURE WORK

Our results explain why the solvers are fast in general, since typically there are few real roots and in general the separation bound is good enough. This agrees with the fact that in most cases the practical complexity of the STURM solver is dominated by the computation of the sequence and not by the evaluation. Our current work extends the first part of this paper to Kac polynomials, and to solvers DESCARTES and BERNSTEIN.

The main issue with the Kac polynomials is that there is a discontinuity at ± 1 when $d \rightarrow \infty$. To be more precise, the fact that there are few roots even near ± 1 , where they are

d	$\sqrt{2d}$	$(-\infty, \infty)$	$(-\infty, -1)$	$(-1, 0)$	$(0, 1)$	$(1, \infty)$
100	14.142	13.640	0.760	2.740	6.530	3.610
150	17.321	16.540	0.890	3.260	8.090	4.300
200	20.000	19.740	1.100	3.780	9.740	5.120
250	22.361	21.400	1.350	3.970	10.610	5.470
300	24.495	24.320	1.270	4.760	12.300	5.990
350	26.458	26.540	1.620	5.100	13.400	6.420
400	28.284	27.980	1.490	5.430	14.080	6.980
450	30.000	29.460	1.620	5.890	14.970	6.980
500	31.623	31.200	1.830	5.960	15.620	7.790
550	33.166	32.740	1.770	6.360	16.290	8.320
600	34.641	34.300	1.850	6.570	17.270	8.610
650	36.056	35.480	2.050	6.840	17.240	9.350
700	37.417	37.200	2.160	7.510	18.650	8.880
750	38.730	38.180	2.190	7.300	19.360	9.330
800	40.000	39.160	2.220	7.830	19.490	9.620
850	41.231	40.420	2.130	8.010	20.320	9.960
900	42.426	41.780	2.390	8.070	20.530	10.790
950	43.589	42.680	2.200	8.330	21.570	10.580
1000	44.721	43.540	2.400	8.610	21.770	10.760

Table 1. Experiments with random polynomial in the Bernstein basis.

concentrated asymptotically, is balanced by the fact the 2-point correlation, $k(s_1, s_2)$, between two consecutive roots is a complicated function of $|s_1 - s_2|$, s_1 and d and (in opposition with the two other distributions we studied) its limit when d tends to infinity is not equivalent to a simple function of $|s_1 - s_2|$. This is an interesting problem which deserves to be studied and investigate further.

An interesting question is whether we can design a randomized exact algorithm based on the properties of random polynomials. Lastly, we wish to extend our study to polynomials with inexact coefficients.

Acknowledgement. We thank the reviewers for their constructive comments. IZE thanks D. Hristopoulos for discussions on the statistics of roots' distributions. AG acknowledges fruitful discussions with Julien Barre. IZE and AG are partially supported by Marie-Curie Network "SAGA", FP7 contract PITN-GA-2008-214584. ET is partially supported by an individual postdoctoral grant from the Danish Agency for Science, Technology and Innovation, and by the State Scholarship Foundation of Greece.

6. REFERENCES

- [1] A. Akritas. An implementation of Vincent's theorem. *Numerische Mathematik*, 36:53–62, 1980.
- [2] D. Armentano and J.-P. Dedieu. A note about the average number of real roots of a Bernstein polynomial system. *J. Complexity*, 25(4):339 – 342, 2009.
- [3] A. T. Bharucha-Reid and M. Sambandham. *Random Polynomials*. Academic Press, 1986.
- [4] P. Bleher and X. Di. Correlations between zeros of a random polynomial. *J. Stat. Physics*, 88(1):269–305, 1997.
- [5] A. Bloch and G. Polya. On the roots of certain algebraic equations. *Proc. London Math. Soc.*, 33:102–114, 1932.
- [6] C.-P. Chen and F. Qi. The best bound in Wallis' inequality. *Proc. AMS*, 133(2):397–401, 2004.
- [7] J. H. Davenport. Cylindrical algebraic decomposition. Technical Report 88–10, School of Mathematical Sciences, University of Bath, England, <http://www.bath.ac.uk/masjhd/>, 1988.
- [8] Z. Du, V. Sharma, and C. K. Yap. Amortized bound for root isolation via Sturm sequences. In D. Wang and L. Zhi, editors, *Int. Workshop on Symbolic Numeric Computing*, pages 113–129, Beijing, China, 2005. Birkhauser.
- [9] A. Edelman and E. Kostlan. How many zeros of a random polynomial are real? *Bulletin AMS*, 32(1):1–37, 1995.
- [10] A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the Descartes method. In *Proc. Annual ACM ISSAC*, pages 71–78, New York, USA, 2006.
- [11] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real Algebraic Numbers: Complexity Analysis and Experimentation. In P. Hertling, C. Hoffmann, W. Luther, and N. Revol, editors, *Reliable Implementations of Real Number Algorithms: Theory and Practice*, volume 5045 of *LNCS*, pages 57–82, 2008.
- [12] P. Erdős and P. Turán. On the distribution of roots of polynomials. *Annals of Mathematics*, 51(1):105–119, 1950.
- [13] J.M. Hammersley. The zeros of a random polynomial. In *Proc. of the 3rd Berkeley Symposium on Mathematical Statistics and Probability*, volume 1955, pages 89–111, 1954.
- [14] J.H. Hannay. Chaotic analytic zero points: exact statistics for those of a random spin state. *J. Physics A: Math. & General*, 29:L101–L105, 1996.
- [15] L. E. Heindel. Integer arithmetic algorithms for polynomial real zero determination. *J. of the Association for Computing Machinery*, 18(4):533–548, October 1971.
- [16] H. Hong. Bounds for absolute positiveness of multivariate polynomials. *J. Symbolic Comp.*, 25(5):571–585, 1998.
- [17] C. P. Hughes and A. Nikeghbali. The zeros of random polynomials cluster uniformly near the unit circle. *Compositio Mathematica*, 144:734–746, Mar 2008.
- [18] J. R. Johnson. *Algorithms for Polynomial Real Root Isolation*. PhD thesis, The Ohio State University, 1991.
- [19] J. R. Johnson, W. Krandick, K. Lynch, D. Richardson, and A. Ruslanov. High-performance implementations of the Descartes method. In *Proc. Annual ACM ISSAC*, pages 154–161, NY, 2006.
- [20] M. Kac. On the average number of real roots of a random algebraic equation. *Bulletin AMS*, 49:314–320 & 938, 1943.
- [21] E. Kowalski. Bernstein polynomials and Brownian motion. *American Mathematical Monthly*, 113(10):865–886, 2006.
- [22] W. Krandick. Isolierung reeller nullstellen von polynomen. In J. Herzberger, editor, *Wissenschaftliches Rechnen*, pages 105–154. Akademie-Verlag, Berlin, 1995.
- [23] T. Lickteig and M-F. Roy. Sylvester-Habicht Sequences and Fast Cauchy Index Computation. *J. Symb. Comput.*, 31(3):315–341, 2001.
- [24] J.E. Littlewood and A.C. Offord. On the number of real roots of a random algebraic equation. *J. London Math. Soc.*, 13:288–295, 1938.
- [25] K. Mehlhorn and S. Ray. Faster algorithms for computing Hong's bound on absolute positiveness. *J. Symbolic Computation*, 45(6):677 – 683, 2010.
- [26] M. Mignotte and D. Ștefănescu. *Polynomials: An algorithmic approach*. Springer, 1999.
- [27] V.Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and rootfinding. *J. Symbolic Computation*, 33(5):701–733, 2002.
- [28] A. Papoulis and S.U. Pillai. *Probability, random variables, and stochastic processes*. McGraw-Hill, 3rd edition, 1991.
- [29] D. Reischert. Asymptotically fast computation of subresultants. In *Proc. Annual ACM ISSAC*, pages 233–240, 1997.
- [30] F. Rouillier and Z. Zimmermann. Efficient isolation of polynomial's real roots. *J. Comput. & Applied Math.*, 162(1):33–50, 2004.
- [31] G. Schehr and S.N. Majumdar. Real roots of random polynomials and zero crossing properties of diffusion equation. *J. Stat. Physics*, 132(2):235–273, 2008.
- [32] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Manuscript. Univ. of Tübingen, Germany, 1982.
- [33] V. Sharma. Complexity of real root isolation using continued fractions. *Theor. Comput. Sci.*, 409(2):292–310, 2008.
- [34] M. Shub and S. Smale. Complexity of bézout's theorem II: volumes and probabilities. In F. Eyssette and A. Galligo, editors, *Computational Algebraic Geometry*, volume 109 of *Progress in Mathematics*, pages 267–285. Birkhäuser, 1993.
- [35] E. P. Tsigaridas and I. Z. Emiris. On the complexity of real root isolation using Continued Fractions. *Theoretical Computer Science*, 392:158–173, 2008.
- [36] C.K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.

The DMM Bound: Multivariate (Aggregate) Separation Bounds

Ioannis Z. Emiris
University of Athens
Athens, Greece
emiris(at)di.uoa.gr

Bernard Mourrain
GALAAD, INRIA Méditerranée
Sophia-Antipolis, France
mourrain(at)inria.fr

Elias P. Tsigaridas
Århus University, Denmark
elias.tsigaridas(at)gmail.com

ABSTRACT

In this paper we derive aggregate separation bounds, named after Davenport-Mahler-Mignotte (DMM), on the isolated roots of polynomial systems, specifically on the minimum distance between any two such roots. The bounds exploit the structure of the system and the height of the sparse (or toric) resultant by means of mixed volume, as well as recent advances on aggregate root bounds for univariate polynomials, and are applicable to arbitrary positive dimensional systems. We improve upon Canny's gap theorem [7] by a factor of $\mathcal{O}(d^{n-1})$, where d bounds the degree of the polynomials, and n is the number of variables. One application is to the bitsize of the eigenvalues and eigenvectors of an integer matrix, which also yields a new proof that the problem is polynomial. We also compare against recent lower bounds on the absolute value of the root coordinates by Brownawell and Yap [5], obtained under the hypothesis there is a 0-dimensional projection. Our bounds are in general comparable, but exploit sparseness; they are also tighter when bounding the value of a positive polynomial over the simplex. For this problem, we also improve upon the bounds in [2, 16]. Our analysis provides a precise asymptotic upper bound on the number of steps that subdivision-based algorithms perform in order to isolate all real roots of a polynomial system. This leads to the first complexity bound of Milne's algorithm [22] in 2D.

Categories and Subject Descriptors

F.2 [Theory of Computation]: Analysis of Algorithms and Problem Complexity; I.1 [Computing Methodology]: Symbolic and algebraic manipulation—*Algorithms*

General Terms

Theory

Keywords

separation bound, polynomial system, mixed volume, Milne, positive polynomial

1. INTRODUCTION

One of the great challenges in algebraic algorithms is to fully understand the theoretical and practical complexity of methods based on exact arithmetic. One goal may be towards hybrid symbolic-numeric approaches that exploit both exact and approximate computations. Computing all roots, in some representation, of systems of multivariate polynomials is a fundamental question in both symbolic and numeric computation. The complexity analysis and the actual runtimes typically depend on *separation bounds*, i.e. the minimum distance between any two, possibly complex, roots of the system. This is particularly true for algorithms based on subdivision techniques and, more generally, for any numerical solver seeking to certify its output. Hence, separation bounds are of great use in areas such as computational geometry and geometric modeling.

Davenport [11] was first to introduce aggregate separation bounds for the real roots of a univariate polynomial, which depend on Mahler's measure, e.g. [20]. Mignotte [21] loosened the hypothesis on the bounds and extended them to complex roots.

As for algebraic systems, a fundamental result is Canny's Gap theorem [7], on the separation bound for square 0-dimensional systems, see Th. 10. Yap [31] relaxed the 0-dimensional requirement by requiring it holds only on the affine part of the variety. A recent lower bound on the absolute value of the root coordinates [5] applies to those coordinates for which the variety's projection has dimension 0, and does not require the system to be square. For arithmetic bounds applied to Nullstellensatz we refer to [18].

Basu, Leroy, and Roy [2] and, recently, Jeronimo and Perucci [16] considered the closely related problem of computing a lower bound for the minimum value of a positive polynomial over the standard simplex. For this, they compute lower bounds on the roots of polynomial system formed by the polynomial and all its partial derivatives. This problem is also treated in [5].

Separation bounds are important for estimating the complexity of subdivision-based algorithms for solving polynomial systems, that depend on exclusion/inclusion predicates or root counting techniques, e.g. [30, 19, 6, 22, 15].

Our contribution. We derive worst-case (aggregate) separation bounds for the roots of polynomial systems, which are not necessarily 0-dimensional. The bounds are computed as a function of the number of variables, the norm of the polynomials, and a bound on the number of roots of well-constrained systems. For the latter we employ mixed volume in order to exploit the sparse structure that appears in many

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

applications. Any future better bound can be used to improve our results. The main ingredients of our proof are resultants, including bounds on their height [28].

We extend the known separation bound for single polynomial equations to 0-dimensional systems, and call it DMM_n , after *Davenport-Mahler-Mignotte*. This improves upon Canny's Gap theorem by $\mathcal{O}(d^{n-1})$. Our bounds are within a factor of $\mathcal{O}(2^n)$ from optimal for certain systems, which is good for n small (or constant) compared to the other parameters. They are comparable to those in [5] on the absolute value of root coordinates, but they are an improvement when expressed using mixed volumes. It seems nontrivial to apply sparse elimination theory to the approach of [5]. More importantly, our result is extended to positive-dimensional systems, thus addressing a problem that has only been examined very recently in [5].

We illustrate our bounds on computing the eigenvalues / eigenvectors of an integer matrix, and improve upon Canny's bound by a factor exponential in matrix dimension. Thanks to mixed volume, we derive a bound polynomial in the logarithm of the input size, hence offering a new alternative to Bareiss' result [1] that the problem is of polynomial bit complexity. We also bound the minimum of a positive polynomial over the standard simplex and improve upon the 3 best known bounds [2, 5, 16], when the total degree is larger than the number of variables.

Finally, we upper bound the number of steps for any subdivision based algorithm using a real-root counter in a box to isolate the real roots of a system in a given domain. This leads to the first complexity bound of Milne's algorithm [22] in $\mathbb{R}^2$. This aggregate separation bound is also useful in the analysis of the subdivision algorithm based on continued fractions expansion [19] for polynomial system solving.

The polynomial systems in practice have a small number of real roots and all roots, real and complex, are well separated; it is challenging to derive an average-case DMM_n . Another open question is to express the positive-dimensional bound wrt the dimension of the excess component.

Paper structure. We introduce some notation, then Sec. 2 derives and proves the multivariate version of DMM as main Thm. 3. Its near-optimality and comparisons to existing bounds are in Sec. 3, which also extends it to positive-dimensional systems. Two applications of our bounds are in Sec. 4. Sec. 5 is devoted to subdivision algorithms.

Notation. $\mathcal{O}$, resp. $\mathcal{O}_B$, means bit, resp. arithmetic, complexity and $\tilde{\mathcal{O}}_B$, resp. $\tilde{\mathcal{O}}$, means we are ignoring logarithmic factors. For a polynomial $f \in \mathbb{Z}[x_1, \dots, x_n]$, where $n \geq 1$, $\deg(f)$ denotes its total degree, while $\deg_{x_i}(f)$ denotes its degree w.r.t. x_i . By $\mathcal{L}(f)$ we denote the maximum bitsize of the coefficients of f (including a bit for the sign). For $a \in \mathbb{Q}$, $\mathcal{L}(a) \geq 1$ is the maximum bitsize of the numerator and denominator. For simplicity, we assume, for any polynomial, $\log(\text{dg}(f)) = \mathcal{O}(\mathcal{L}(f))$. Let $\text{sep}(f)$, resp. $\text{sep}(\Sigma)$, denote the separation bound, i.e. the minimum distance between two, possibly complex, roots of polynomial f , resp. system (Σ) . For $f = a_d \prod_{i=1}^d (x - z_i) \in \mathbb{C}[x]$, with $a_d \neq 0$, its Mahler measure is $\mathcal{M}(f) := 4|a_d| \prod_{i=1}^d \max\{1, |z_i|\}$.

2. THE DMM BOUND

The univariate case. Consider a real univariate polynomial A , not necessarily square-free, of degree d and its complex roots γ_j in ascending magnitude, where $j \in \{1, 2, \dots, d\}$. The next theorem [29] bounds the product of differences of

the form $|\gamma_i - \gamma_j|$. It slightly generalizes a theorem in [20], which in turn generalizes [11], see also [17, 13].

Theorem 1 (DMM₁). *Let $f \in \mathbb{C}[X]$, with $\deg(f) = d$ and not necessarily square-free. Let Ω be any set of ℓ couples of indices (i, j) , $1 \leq i < j \leq d$, and let the distinct non-zero (complex) roots of f be $0 < |\gamma_1| \leq |\gamma_2| \leq \dots \leq |\gamma_d|$. Then*

$$2^\ell \mathcal{M}(f)^\ell \geq \prod_{(i,j) \in \Omega} |\gamma_i - \gamma_j| \geq 2^{\ell - \frac{d(d-1)}{2}} \mathcal{M}(f)^{1-d-\ell} \sqrt{|\text{disc}(f_{\text{red}})|},$$

where f_{red} is the square-free part of f . If $f \in \mathbb{Z}[x]$, $\ell \leq d$ and $\mathcal{L}(f) = \tau$, then

$$\begin{aligned} d^{d/2} 2^{2d\tau} &\geq d^{\ell/2} 2^{2\ell\tau} \geq \prod_{(i,j) \in \Omega} |\gamma_i - \gamma_j| \\ &\geq d^{-d} 2^{-d^2 - 3\tau(\ell+d)} \geq d^{-d} 2^{-d^2 - 6d\tau}. \end{aligned}$$

The second inequality follows from: $\mathcal{M}(f) \leq \|f\|_2 \leq (d+1)\|f\|_\infty \leq (d+1)^{\frac{1}{2}} 2^\tau$, e.g. [20, 31]. In the first inequality we can replace $\mathcal{M}(f)$ by $\|f\|_2$.

The bound of Thm. 1 has an additional factor of 2^{d^2} wrt [11, 13], which is, asymptotically, not significant when the polynomial is not square-free or $d = \mathcal{O}(\tau)$. The current version of the theorem has very loose hypotheses and applies to non-squarefree polynomials.

Roughly, DMM_1 provides a bound on all distances between consecutive roots of a polynomial. This quantity is, asymptotically, almost equal to the separation bound. The interpretation is that not all roots of a polynomial can be very close together or, quoting J.H. Davenport, “not all [distances between the roots] could be bad”.

The multivariate case. This section generalizes DMM_1 to 0-dimensional polynomial systems. Let $n > 1$ be the number of variables. We use $\mathbf{x}^e$ to denote the monomial $x_1^{e_1} \dots x_n^{e_n}$, with $\mathbf{e} = (e_1, \dots, e_n) \in \mathbb{Z}^n$. The input is *Laurent polynomials* $f_1, \dots, f_n \in K[x_1^\pm, \dots, x_n^\pm] = K[\mathbf{x}, \mathbf{x}^{-1}]$, where $K \subset \mathbb{C}$ is the coefficient field. Since we can multiply Laurent polynomials by monomials without affecting their nonzero roots, in the sequel we assume there are no negative exponents. Let the polynomials be

$$f_i = \sum_{j=1}^{m_i} c_{i,j} \mathbf{x}^{a_{i,j}}, \quad 1 \leq i \leq n. \quad (1)$$

Let $\{a_{i,1}, \dots, a_{i,m_i}\} \subset \mathbb{Z}^n$ be the support of f_i ; its Newton polytope Q_i is the convex hull of the support. Let $\text{MV}(Q_1, \dots, Q_n) > 0$ be the *mixed volume* of convex polytopes $Q_1, \dots, Q_n \subset \mathbb{R}^n$. Here is Bernstein's bound, known also as BKK bound.

Theorem 2. *For $f_1, \dots, f_n \in \mathbb{C}[\mathbf{x}, \mathbf{x}^{-1}]$ with Newton polytopes $Q_1, \dots, Q_n$, the number of common isolated solutions in $(\mathbb{C}^*)^n$, multiplicities counted, does not exceed $\text{MV}(Q_1, \dots, Q_n)$, independently of the corresponding variety's dimension.*

We consider polynomial system

$$(\Sigma) : f_1(\mathbf{x}) = f_2(\mathbf{x}) = \dots = f_n(\mathbf{x}) = 0, \quad (2)$$

where $f_i \in \mathbb{R}[\mathbf{x}^{\pm 1}]$, which we assume to be 0-dimensional. We are interested in its roots in $(\mathbb{C}^*)^n$, which are called toric. We denote by Q_0 the convex hull of the unit standard simplex. Let $M_i = \text{MV}(Q_0, \dots, Q_{i-1}, Q_{i+1}, \dots, Q_n)$, and $\#Q_i$ denote the number of lattice points in the closed polytope Q_i . Wlog, assume $\dim \sum_{i=0}^n Q_i = n$ and $\dim \sum_{i \in I} Q_i \geq j$

for any $I \subset \{0, \dots, n\}$ with $|I| = j$. We consider the *sparse* (or *toric*) resultant of a system of $n+1$ polynomial equations in n variables, assuming we have fixed the $n+1$ supports. It provides a condition on the coefficients for the solvability of the system, and generalizes the classical resultant of n homogeneous polynomials, by taking into account the supports of the polynomials. For details, see [9].

Let D be the number of roots $\in (\mathbb{C}^*)^n$ of (Σ) , multiplicities counted, so $D \leq M_0$. We also use $B = (n-1) \binom{D}{2}$, and $\text{dg}(f_i) = d_i \leq d$. If $f_i \in \mathbb{Z}[\mathbf{x}^{\pm 1}]$, $\mathcal{L}(f_i) = \tau_i \leq \tau$, $1 \leq i \leq n$. Now $\text{vol}(\cdot)$ stands for Euclidean volume, and $(\#Q_i)$ for the number of lattice points in Q_i ; the inequality connecting $(\#Q_i)$ and polytope volume is in [4]. We present the abbreviations and inequalities used throughout the paper:

$$\begin{aligned} D &\leq M_0 \leq \prod_{i=1}^n d_i \leq d^n, \quad B \leq nD^2 \leq n \prod_{i=1}^n d_i^2 \leq nd^{2n}, \\ M_i &\leq \prod_{\substack{1 \leq j \leq n \\ j \neq i}} d_j = D_i, \quad \sum_{i=1}^n M_i \leq \sum_{i=1}^n D_i \leq nd^{n-1}, \\ (\#Q_i) &\leq n! \text{vol}(Q_i) + n \leq d_i^n + n \leq 2d_i^n, \\ A &= \prod_{i=1}^n \sqrt{M_i} 2^{M_i} \leq 2^{nd^{n-1} + \frac{n^2-n}{2} \lg d}, \\ C &= \prod_{i=1}^n \|f_i\|_{\infty}^{M_i} \leq 2^{\tau \sum_{i=1}^n M_i} \leq 2^{n\tau d^{n-1}}, \\ h &\leq (n+1)^D \varrho \leq (n+1)^{d^n} 2^{nd^{n-1}} d^{n^2 d^{n-1}}, \\ \varrho &= \prod_{i=1}^n (\#Q_i)^{M_i} \leq 2^{\sum_{i=1}^n D_i} \prod_{i=1}^n d_i^{n D_i} \leq 2^{nd^{n-1}} d^{n^2 d^{n-1}}. \end{aligned} \quad (3)$$

Theorem 3 (DMM_n). Consider the 0-dimensional polynomial system (Σ) in (2). Let D be the number of complex solutions of the system in $(\mathbb{C}^*)^n$, which are $0 < |\gamma_1| \leq |\gamma_2| \leq \dots \leq |\gamma_D|$. Let Ω be any set of ℓ couples of indices (i, j) such that $1 \leq i < j \leq D$ and $\gamma_i \neq \gamma_j$. Then the following holds

$$(2^{D+1} \varrho C)^{\ell} \geq \prod_{(i,j) \in \Omega} |\gamma_i - \gamma_j| \geq 2^{-\ell - (D-1)(D+2)/2} (hC)^{1-D-\ell} B^{(1-n)(D^2+D(\ell-1)+\ell)} \sqrt{|U_r|}, \quad (4)$$

where $|U_r|$ denotes the discriminant of the square-free part of the u -resultant, and $|\cdot|$ denotes absolute value. If $f_i \in \mathbb{Z}[\mathbf{x}]$ and $\gamma_{j,k}$ stands for the k -th coordinate, $1 \leq k \leq n$, of γ_j , then:

$$(2^D \varrho C)^{-1} \leq |\gamma_{j,k}| \leq 2^D \varrho C, \quad (5)$$

$$\text{sep}(\Sigma) \geq 2^{-(3D+2)(D-1)/2} (\sqrt{D+1} \varrho C)^{-D}. \quad (6)$$

The following corollary employs mixed volumes.

Corollary 4. Under the hypothesis of Th. 3, for $f_i \in \mathbb{Z}[\mathbf{x}^{\pm 1}]$, $i = 1, \dots, n$, we have

$$\begin{aligned} 2^{M_0(1+M_0+\sum_{i=1}^n M_i(\tau+\lg(\#Q_i)))} &\geq \prod_{(i,j) \in \Omega} |\gamma_i - \gamma_j| \geq \\ 2^{-2M_0 \sum_{i=1}^n M_i(\tau+\lg(\#Q_i)) - 2M_0^2(1+\lg(n+1)+n \lg(n)+2n \lg(M_0))}, & \\ 2^{-(M_0+\sum_{i=1}^n M_i(\tau+\lg(\#Q_i)))} \leq |\gamma_{j,k}| \leq 2^{M_0+\sum_{i=1}^n M_i(\tau+\lg(\#Q_i))} & \end{aligned} \quad (7)$$

$$\text{sep}(\Sigma) \geq 2^{-M_0(\frac{3}{2}M_0+\lg(M_0)+\sum_{i=1}^n M_i(\tau+\lg(\#Q_i)))}. \quad (8)$$

Corollary 5. Under the hypothesis of Th. 3, for $f_i \in \mathbb{Z}[\mathbf{x}^{\pm 1}]$, $\text{dg}(f_i) \leq d$ and $\mathcal{L}(f_i) \leq \tau$, we have

$$\prod_{(i,j) \in \Omega} |\gamma_i - \gamma_j| \geq 2^{-(3+4 \lg n + 4n \lg d)d^{2n}} 2^{-2n(1+n \lg d + \tau)d^{2n-1}}, \quad (9)$$

$$2^{-d^n - n(\tau+n \lg d+1)d^{n-1}} \leq |\gamma_{j,k}| \leq 2^{d^n + n(\tau+n \lg d+1)d^{n-1}}, \quad (10)$$

$$\text{sep}(\Sigma) \geq 2^{-2d^{2n} - n(2n \lg d + \tau)d^{2n-1}}. \quad (11)$$

Proof of main theorem. Let us first establish the lower bound. Let $\gamma_i = (\gamma_{i,1}, \dots, \gamma_{i,n}) \in (\mathbb{C}^*)^n$, $1 \leq i \leq D$, be the solutions of (Σ) , where f_i are defined in (1). We denote the set of solutions as $V \subset (\mathbb{C}^*)^n$. We add an equation to (Σ) to obtain:

$$(\Sigma_0) : f_0(\mathbf{x}) = f_1(\mathbf{x}) = \dots = f_n(\mathbf{x}) = 0, \quad (12)$$

where

$$f_0 = u + r_1 x_1 + r_2 x_2 + \dots + r_n x_n, \quad (13)$$

$r_1, \dots, r_n \in \mathbb{Z}$ to be defined in the sequel, and u is a new parameter. Now $u = -\sum_i r_i \gamma_{j,i}$, on a solution γ_j . We choose properly the coefficients of f_0 to ensure that the function

$$f_0 : V \rightarrow \mathbb{C}^* : \gamma \mapsto f_0(\gamma)$$

is injective. The separating element shall ensure injectivity [3, 7, 14, 27].

Proposition 6. Let $V \subset \mathbb{C}^n$ with cardinality D . The set of linear forms

$$\mathcal{F} = \{u_i = x_1 + i x_2 + \dots + i^{n-1} x_n \mid 0 \leq i \leq B = (n-1) \binom{D}{2}\}$$

contains at least one separating element, which takes distinct values on V .

Corollary 7. For $f_0 \in \mathcal{F}$ it holds that $\|f_0\|_{\infty} \leq B^{n-1}$, and

$$\|f_0\|_{\infty} \leq \|f_0\|_2 \leq 2B^{n-1} = 2(n-1)^{n-1} \left(\frac{D}{2}\right)^{n-1}.$$

Proof: The first inequality is evident from the definition of infinite norm. For the second inequality, $B = (n-1) \binom{D}{2}$:

$$\begin{aligned} \|f_0\|_{\infty} &\leq \|f_0\|_2 \leq \sqrt{1 + B^2 + B^4 + \dots + (B^2)^{n-1}} \\ &\leq \sqrt{\frac{B^{2n}-1}{B^2-1}} \leq \sqrt{\frac{B^{2n-2}}{1-1/B^2}} \leq \sqrt{4B^{2n-2}} = 2B^{n-1}. \end{aligned}$$

□

We consider the u -resultant U of (Σ_0) that eliminates $\mathbf{x}$. It is univariate in u , with coefficients homogeneous polynomials in the coefficients of (Σ_0) , e.g. [9]:

$$U(u) = \dots + \varrho_k u^k \mathbf{r}_k^{D-k} \mathbf{c}_{1,k}^{M_1} \mathbf{c}_{2,k}^{M_2} \dots \mathbf{c}_{n,k}^{M_n} + \dots, \quad (14)$$

where $\varrho_k \in \mathbb{Z}$, $\mathbf{c}_{j,k}^{M_j}$ denotes a monomial in coefficients of f_j with total degree M_j , and $\mathbf{r}_k^{D-k}$ denotes a monomial in the coefficients of f_0 of total degree $D-k$. The degree of U , with respect to u is D . It holds that

$$\left| \mathbf{c}_{1,k}^{M_1} \mathbf{c}_{2,k}^{M_2} \dots \mathbf{c}_{n,k}^{M_n} \right| \leq C = \prod_{i=1}^n \|f_i\|_{\infty}^{M_i}, \quad (15)$$

From Cor. 7 we have that $|\mathbf{r}_k| \leq \|f_0\|_\infty \leq B^{n-1}$, for all k . Let $|\varrho_k| \leq h$, for all k . Then using [28], see also Eq. (4), we get that

$$h \leq \prod_{i=0}^n (\#Q_i)^{M_i} = (\#Q_0)^D \prod_{i=1}^n (\#Q_i)^{M_i} = (n+1)^D \varrho.$$

We can bound the norm of U :

$$\begin{aligned} \|U\|_2^2 &\leq \sum_{k=0}^D \left| \varrho_k \mathbf{r}_k^{D-k} \mathbf{c}_{1,k}^{M_1} \mathbf{c}_{2,k}^{M_2} \dots \mathbf{c}_{n,k}^{M_n} \right|^2 \\ &\leq \sum_{k=0}^D |h (B^{n-1})^{D-k} C|^2 \leq h^2 C^2 \sum_{k=0}^D (B^{2n-2})^{D-k} \\ &\leq h^2 C^2 \sum_{k=0}^D (B^{2n-2})^k \leq h^2 C^2 \frac{(B^{2n-2})^{D+1} - 1}{B^{2n-2} - 1} \\ &\leq h^2 C^2 4 (B^{2n-2})^D \leq 4 h^2 C^2 B^{2(n-1)D}, \end{aligned}$$

and so

$$\|U\|_\infty \leq \|U\|_2 \leq 2 h C B^{(n-1)D} \leq 2(n+1)^D \varrho C B^{(n-1)D}.$$

If u_j are the distinct roots of U , then by recalling the injective nature of f_0 , we deduce that $u_j = -\sum_{i=1}^n r_i \gamma_{j,i}$. Actually the u -resultant is even stronger, since the multiplicities of its roots correspond to the multiplicities of the solutions of the system, but we will not exploit this further.

Proposition 8 (Cauchy-Bunyakovsky-Schwartz). *Let $a_1, a_2, \dots, a_n \in \mathbb{C}$ and $b_1, b_2, \dots, b_n \in \mathbb{C}$. Then,*

$$|\bar{a}_1 b_1 + \dots + \bar{a}_n b_n|^2 \leq (|a_1|^2 + \dots + |a_n|^2) (|b_1|^2 + \dots + |b_n|^2),$$

where $\bar{a}_i$ denotes the complex conjugate of a_i , and $1 \leq i \leq n$. Equality holds if, for all i , $a_i = 0$ or if there is a scalar λ such that $b_i = \lambda a_i$.

Consider γ_i, γ_j and let u_i, u_j be the corresponding roots of U . Using Prop. 8,

$$\begin{aligned} &|r_1(\gamma_{i,1} - \gamma_{j,1}) + \dots + r_n(\gamma_{i,n} - \gamma_{j,n})|^2 \\ &\leq (r_1^2 + \dots + r_n^2) (|\gamma_{i,1} - \gamma_{j,1}|^2 + \dots + |\gamma_{i,n} - \gamma_{j,n}|^2) \Leftrightarrow \end{aligned}$$

$$\begin{aligned} &|\sum_{k=1}^n r_k \gamma_{i,k} - \sum_{k=1}^n r_k \gamma_{j,k}|^2 \leq \sum_{k=1}^n r_k^2 \cdot \sum_{k=1}^n |\gamma_{i,k} - \gamma_{j,k}|^2 \\ &\Leftrightarrow |u_i - u_j|^2 \leq (\sum_{k=1}^n r_k^2) \cdot |\gamma_i - \gamma_j|^2, \end{aligned}$$

and thus

$$|\gamma_i - \gamma_j| \geq \left(\sum_{k=1}^n r_k^2 \right)^{-1/2} |u_i - u_j|.$$

To prove the lower bound of Th. 3, we apply the previous inequality for all pairs in Ω , $|\Omega| = \ell$. So we get

$$\prod_{(i,j) \in \Omega} |\gamma_i - \gamma_j| \geq \left(\sum_{k=1}^n r_k^2 \right)^{-\frac{1}{2}\ell} \prod_{(i,j) \in \Omega} |u_i - u_j|. \quad (16)$$

It remains to bound the two factors of RHS of the previous inequality. To bound the first we use Cor. 7. It holds $\sum_{k=1}^n r_k^2 \leq 1 + \sum_{k=1}^n r_k^2 \leq \|f_0\|_2^2 \leq 4 B^{2n-2}$, so

$$\left(\sum_{k=1}^n r_k^2 \right)^{-\frac{1}{2}\ell} \geq 2^{-\ell} B^{(1-n)\ell}. \quad (17)$$

For the second factor of (16) we apply DMM₁ to U ; and thus

$$\begin{aligned} \prod_{(i,j) \in \Omega} |u_i - u_j| &\geq 2^{\ell-D(D-1)/2} \|U\|_2^{1-D-\ell} \sqrt{|U_r|} \\ &\geq 2^{(1-D)(D+2)/2} (h C B^{(n-1)D})^{1-D-\ell} \sqrt{|U_r|}. \end{aligned} \quad (18)$$

Combining (16) with (17) and (18), we have the lower bound. In the case where the polynomials are in $\mathbb{Z}[\mathbf{x}]$, then it holds that the absolute value of the discriminant of a square-free polynomial is ≥ 1 , and we can omit it from the inequality. If the polynomials are in $\mathbb{Q}[\mathbf{x}]$ the bounds are almost the same, since they depend on Mahler's measure.

Let us now establish the upper bound. We specialize f_0 in (13) by setting $r_i = -1$, for some $i \in \{1, \dots, n\}$, and $r_j = 0$, where $1 \leq j \leq n$ and $j \neq i$. Wlog assume $r_1 = -1$. We compute the u -resultant of the system, which we call $\mathcal{R}_1 \in \mathbb{Z}[u]$. Its roots are the first coordinates of the isolated zeros of the system, viz. $\gamma_{1,i}$, $1 \leq i \leq D$. Thus $\text{dg}(\mathcal{R}_1) \leq D$.

The coefficients of $\mathcal{R}_1$ are of the form, $\varrho_k \mathbf{c}_1^{M_1} \mathbf{c}_2^{M_2} \dots \mathbf{c}_n^{M_n}$, where $\varrho_k \in \mathbb{Z}$ and the interpretation of the rest of the formula is the same as in the previous section. Using [28], see also Eq. (4), we get that

$$|\varrho_k| \leq \prod_{i=0}^n (\#Q_i)^{M_i} = (\#Q_0)^D \prod_{i=1}^n (\#Q_i)^{M_i} = 2^D \varrho,$$

since now f_0 is a simplex in dimension 1. It also holds that $|\mathbf{c}_1^{M_1} \mathbf{c}_2^{M_2} \dots \mathbf{c}_n^{M_n}| \leq C$. Combining the two inequalities we deduce that

$$\|\mathcal{R}_1\|_\infty \leq 2^D \varrho C,$$

and also $\|\mathcal{R}_1\|_\infty \leq \|\mathcal{R}_1\|_2 \leq 2^D \sqrt{D+1} \varrho C$.

From Cauchy's bound for the roots of univariate polynomials, e.g. [20], we know that for all the roots of $\mathcal{R}_1$ it holds that $(2^D \varrho C)^{-1} \leq 1/\|\mathcal{R}_1\|_\infty \leq |\gamma_{i,j}| \leq \|\mathcal{R}_1\|_\infty \leq 2^D \varrho C$. The inequality holds for all the indices i and j . Hence, all roots of the system in $(\mathbb{C}^*)^n$ are contained in a high-dimensional annulus in $\mathbb{C}^n$, defined as the difference of the volumes of two spheres centered at the origin, with radii $2^D \varrho C$ and $(2^D \varrho C)^{-1}$, resp. This proves Eq. (5).

Now we are ready to prove the upper bound of Eq. (4) in Th. 3. For all $a, b \in \mathbb{C}$ it holds that

$$|a - b| \leq 2 \max\{|a|, |b|\}. \quad (19)$$

Let the multiset $\bar{\Omega} = \{j \mid (i, j) \in \Omega\}$, where $|\bar{\Omega}| = \ell$, then

$$\prod_{(i,j) \in \Omega} |\gamma_i - \gamma_j| \leq 2^\ell \prod_{j \in \bar{\Omega}} |\gamma_j| \leq 2^\ell (2^D \varrho C)^\ell \leq (2^{D+1} \varrho C)^\ell.$$

For proving (6), let (i, j) be the pair of indices where the separation bound of (Σ) is attained. Then

$$\text{sep}(\Sigma) = |\gamma_i - \gamma_j| = \sqrt{\sum_{k=1}^n (\gamma_{i,k} - \gamma_{j,k})^2} \geq |\gamma_{i,1} - \gamma_{j,1}| \geq \text{sep}(\mathcal{R}_1),$$

where k is any index such that $\gamma_{i,k} \neq \gamma_{j,k}$ and $\text{sep}(\mathcal{R}_1)$ is the separation bound of $\mathcal{R}_1$. An easy bound on the latter can be derived by applying Th. 1 to $\mathcal{R}_1$ with $\ell = 1$: $\text{sep}(\mathcal{R}_1) \geq 2^{1-(\frac{D}{2})} \|\mathcal{R}_1\|_2^{-D} \geq 2^{(1-D)(D+2)/2} (D+1)^{-D/2} (2^D \varrho C)^{-D} \geq 2^{-(3D+2)(D-1)/2} (\sqrt{D+1} \varrho C)^{-D}$, which completes the proof of (6).

Remark 9. *It is tempting to try to prove the lower bound of Th. 3 by applying DMM₁ to $\mathcal{R}_1$, instead of U , as we did in the previous section. This would allow us to eliminate the factor $B^{-(n-1)(D^2+\ell(D+1)-D)}$ from the result. However, if we apply DMM₁ to $\mathcal{R}_1$, it is not obvious that the requirements of Th. 1 hold, i.e. that the ordering of (the coordinates of) the roots is preserved. Moreover, the bounds on the u -resultant are of independent interest, since the latter is used in many algorithms for system solving, e.g. [3, 14, 27].*

3. COMPARISONS AND EXTENSIONS

One of the first multivariate separation bounds was due to Canny, later generalized to the case when only the affine part of the variety is 0-dimensional [31].

Theorem 10 (Gap theorem). [7] *Let $f_1(\mathbf{x}), \dots, f_n(\mathbf{x})$ be polynomials of degree d and coefficient magnitude c , with finitely-many common solutions when homogenized. If $\gamma_j \in \mathbb{C}^n$ is such a solution, then for any k , either $\gamma_{j,k} = 0$ or $|\gamma_{j,k}| > (3dc)^{-n d^n}$.*

Let $\mathcal{L}(f_i) = \tau$, then this becomes $2^{(\lg 3 + \lg d + \tau)nd^n}$, which is worse than the bound in Eq. (10), by a factor of $O(d^{n-1})$. In [5], they only require that the system has a 0-dimensional projection; m is the number of polynomials and $b < n$ the dimension of the prime component where the 0-dimensional projection is considered. The bound is:

$$|\gamma_{ij}| \geq ((n+1)^2 e^{n+2})^{-(n+1)d^n} (b^{n-b-1} m 2^\tau)^{-(n-b)d^{n-b-1}},$$

This is similar to ours in (5), and we make a comparison in the sequel. Moreover, Cor. 4 does not depend on the (total) degree of the equations, but rather on mixed volume, which is advantageous for sparse systems.

A natural question is how close are the bounds to optimum. Let us consider the following system [7]:

$$2^\tau x_1^2 = x_1, x_j = x_{j-1}^d, 2 \leq j \leq n.$$

The roots are $x_j = (2^{-\tau})^{d^{j-1}}$, for $2^\tau \gg 1$. Th. 3 implies $x_j \geq 2^{-d^n - n(\tau + n \lg d + 1)d^{n-1}}$, which, if $\tau \gg d$, is off only by a factor of 2^n asymptotically. The negative exponent of our bound is $\mathcal{O}(n(n \lg d + \tau)d^{n-1})$, Canny's bound gives a negative exponent of $(\lg 3 + \lg d + \tau)nd^n = \mathcal{O}(n\tau d^n)$. The bound from [5] has negative exponent: $n(n+1)(2 \lg(n+1) + n+2)d^n + n(\lg n + \tau)d^{n-1} = \mathcal{O}(n^3 d^n + n\tau d^{n-1})$.

We now consider the case that (Σ) is not 0-dimensional. Then, the bounds of Th. 3 do not hold because they are based on bounding the infinite norm of the u -resultant, which is identically zero. Specifically, the (sparse) resultant vanishes identically when the specialized coefficients of the polynomials are not generic enough, i.e. the variety has positive dimension, or, simply, if the variety has a component of positive dimension at infinity, known as excess component.

To overcome the latter, Canny introduced the Generalized Characteristic Polynomial (GCP) [8] for dense systems. We use its generalization, called Toric GCP (TGCP) [10]. We consider (Σ_0) in (12) and perturb it:

$$(\tilde{\Sigma}_0) \quad \begin{cases} \tilde{f}_0 = f_0 = 0, \\ \tilde{f}_i = f_i + p_i = 0, \quad 1 \leq i \leq n, \end{cases}$$

where $p_i = \sum_{\mathbf{a} \in \mathcal{D}_i} s^{\omega_i(\mathbf{a})} \mathbf{x}^{\mathbf{a}}$, $\omega_i(\cdot)$ are (suitable) linear forms, s a new parameter, and $\mathcal{D}_i$ is the subset of vertices in Q_i corresponding to monomials of f_i on the diagonal of some sparse resultant matrix; at worst, $\mathcal{D}_i$ contains the vertices of Q_i . This perturbation does not alter the support of the polynomials nor the mixed volume of the system.

The TGCP is the sparse resultant of $(\tilde{\Sigma}_0)$, denoted $T \in (\mathbb{Z}[\mathbf{c}, \mathbf{r}])[u, s]$, where $\mathbf{c}$ corresponds to the coefficients of f_i and $\mathbf{r}$ to the coefficients of f_0 . The lowest-degree nonzero coefficient of T , seen as univariate polynomial in s , is a projection operator: it vanishes on the projection of any 0-dimensional component of the algebraic set defined by (Σ_0) . We call this $T_U \in \mathbb{Z}[\mathbf{c}, \mathbf{r}][u]$, and $\text{dg}(T_U) \leq M_0$. The

roots of T_U are the isolated points of the variety plus some points embedded in its positive-dimensional components. It remains to bound the coefficients of T_U . Repeating the construction of U in Eq. (14), we get

$$T_U = \dots + \underbrace{\varrho_k u^k \mathbf{r}_k^{M_0-k} \tilde{\mathbf{c}}_{1,k}^{M_1} \tilde{\mathbf{c}}_{2,k}^{M_2} \dots \tilde{\mathbf{c}}_{n,k}^{M_n}}_{t_k} + \dots,$$

where $\rho_k \in \mathbb{Z}$, and $\tilde{\mathbf{c}}_{i,k}^{M_i}$ is a monomial in the coefficients c_{ij} , s , of total degree M_i . It is an overestimation, wrt the height of T , if we suppose that $\tilde{\mathbf{c}}_{i,k}$ is obtained by adding s^λ to each coefficient of $\mathbf{c}_{i,k}$, where $\lambda = \max_{i,\mathbf{a}} \{\omega_i(\mathbf{a})\}$. If we expand $\tilde{\mathbf{c}}_{i,k}^{M_i}$, the absolute value of the coefficients of s is bounded by $\binom{M_i}{M_i/2} \|f_i\|_\infty^{M_i} \leq 2^{M_i} \|f_i\|_\infty^{M_i} / \sqrt{M_i}$. If we expand the term t_k of T , the degree of s is bounded by $\lambda \cdot \prod_{i=1}^n M_i$, and the coefficients are bounded by

$$\prod_{i=1}^n M_i \cdot |\varrho_k| \cdot |\mathbf{r}_k|^{M_0-k} \cdot \prod_{i=1}^n 2^{M_i} \|f_i\|_\infty^{M_i} / \sqrt{M_i} = |\varrho_k| \cdot |\mathbf{r}_k|^{M_0-k} \cdot \prod_{i=1}^n \sqrt{M_i} \cdot 2^{M_i} \cdot \|f_i\|_\infty^{M_i} = |\mathbf{r}_k|^{M_0-k} h A C,$$

since every factor $\tilde{\mathbf{c}}_{i,k}^{M_i}$ contributes at most M_i coefficients. The bound holds for (the absolute of) all the coefficients of T if we consider it as bivariate polynomial in s, u . Recall that $|\varrho_k| \leq h$, for all k , where h is defined in Eq. (4). This expression also defines A, C .

Now $k \leq M_0$. If we consider T_U as a univariate polynomial in s , then its coefficients are univariate polynomials in u , with degree $\leq M_0$. For the 2-norm of T_U , we use a summation as in the 0-dimensional case, and get

$$\|T_U\|_\infty \leq \|T_U\|_2 \leq 2 h A C B^{(n-1)M_0}.$$

The previous bound is the one on U multiplied by A . Thus we can provide a theorem extending Th. 3 to positive-dimensional systems, by replacing C by AC , in Th. 3,

Theorem 11 (DMM_n with excess components).

Consider the polynomial system (Σ) in (2), which is not necessarily 0-dimensional, and where it holds that $f_i \in \mathbb{Z}[\mathbf{x}]$, $\text{dg}(f_i) \leq d$, and $\mathcal{L}(f_i) \leq \tau$. Let D be the number of the isolated points of the solution set in $(\mathbb{C}^*)^n$, which are $0 < |\gamma_1| \leq |\gamma_2| \leq \dots \leq |\gamma_D|$. Let Ω be any set of ℓ couples of indices (i, j) such that $1 \leq i < j \leq D$, and $\gamma_{j,k}$ stands for the k -th coordinate of γ_j . Then the following holds

$$(2^{M_0+1} \varrho C A)^\ell \geq \prod_{(i,j) \in \Omega} |\gamma_i - \gamma_j| \geq 2^{-\ell - (M_0-1)(M_0+2)/2} (h C A)^{1-M_0-\ell} B^{(1-n)(M_0^2+M_0(\ell-1)+\ell)},$$

$$(2^{M_0} \varrho C A)^{-1} \leq |\gamma_{j,k}| \leq 2^{M_0} \varrho C A, \quad (20)$$

$$\text{sep}(\Sigma) \geq 2^{-(3M_0+2)(M_0-1)/2} (\sqrt{M_0+1} \varrho C A)^{-M_0},$$

We also have the following, less accurate bounds:

$$\prod_{(i,j) \in \Omega} |\gamma_i - \gamma_j| \geq 2^{-(n^2-n)d^n \lg \sqrt{d} - (3+4 \lg n + 4n \lg d)d^{2n}} \times 2^{-2n(2+n \lg d + \tau)d^{2n-1}},$$

$$2^{(n^2-n) \lg \sqrt{d} - d^n - n(\tau + n \lg d + 2)d^{n-1}}$$

$$\leq |\gamma_{j,k}| \leq 2^{(n^2-n) \lg \sqrt{d} + d^n + n(\tau + n \lg d + 2)d^{n-1}},$$

$$\text{sep}(\Sigma) \geq 2^{-(n^2-n)d^n \lg \sqrt{d} - 2d^{2n} - n(2n \lg d + \tau + 1)d^{2n-1}}.$$

4. APPLICATIONS

We illustrate the bounds of Th. 3 in two applications. The first concerns matrix eigenvalues and eigenvectors, and is a standard illustration of the superiority of mixed volumes against Bézout's bound. The second is lower bounds of positive multivariate polynomials, inspired by [2].

Eigenvalues and eigenvectors. Consider an $n \times n$ integer matrix A , with elements $\leq 2^\tau$. We are interested in its eigenvalues λ , and its eigenvectors $\mathbf{v} = (v_1, \dots, v_n)^\top$. This is equivalent to solving $f_j = \sum_{i=1}^n a_{i,j} v_i - \lambda v_j$, $1 \leq i \leq n$, $1 \leq j \leq n$, and $f_{n+1} = \sum_{i=1}^n v_i^2 - 1$. We have $\|f_j\|_\infty \leq 2^\tau$, $\|f_{n+1}\|_\infty \leq 2$. The Bézout bound is 2^{n+1} , whereas the actual number of (complex) solutions is $2n$, which equals the mixed volume, e.g. [14].

Canny's Gap theorem [7] implies $|z| > (6 \cdot 2^\tau)^{-(n+1)2^n}$, for any eigenvalue or eigenvector element $z \neq 0$. Thus, we need $\mathcal{O}(n\tau 2^n)$ bits. We get the same exponential behavior in n if we apply [31] or [5].

It is reasonable to assume that the system is 0-dimensional and apply (5) of Th. 3. It holds that $M_j = 2n$, $M_{n+1} = n$, $(\#Q_{n+1}) \leq 2^{n+2}$, and $(\#Q_i) \leq 2^{n+2}$ where $1 \leq j \leq n$, and $C = \|f_{n+1}\|_\infty^{M_{n+1}} \prod_{j=1}^n \|f_j\|_\infty^{M_j} \leq 2^{\tau \sum_{j=1}^n M_j} 2^n = 2^{2n^2\tau+n}$, $\varrho \leq \prod_{i=1}^{n+1} (\#Q_i)^{M_i} \leq (\#Q_{n+1})^{M_{n+1}} \prod_{i=1}^n (\#Q_i)^{M_i}$; hence $\varrho \leq (2^{n+2})^n \prod_{i=1}^n (2^{n+2})^{2n} \leq 2^{2n^3+5n^2+2n}$.

The solutions lie in $\mathbb{C}^{n+1}$. The lower bound of Th. 3 yields

$$|z| > 2^{-2n^3-5n^2-5-2n^2\tau},$$

where z is an eigenvalues or element of eigenvector. This is exponentially better than the previous bounds. Eq. (6) from Th. 3 bounds the system's separation bound: $-\lg(\text{sep}(\Sigma)) \leq 4n^3\tau + n \lg n + 4n^4 + 10n^3 + 12n^2 + n - 1 = \mathcal{O}(n^4 + n^3\tau)$. This is polynomial in the size of the input, and hence we obtain a new proof of Bareiss' result [1], that computing the eigenvalues and eigenvectors of an integer matrix is a polynomial problem.

Positive multivariate polynomials. We consider the following problem, studied in [2]. Let $P \in \mathbb{Z}[x_1, \dots, x_n]$ be a multivariate polynomial of degree d which on the n -dimensional simplex takes only positive values. We are interested in computing a bound on its *minimum value*, m . We may assume that the minimum is attained inside the simplex; if not, apply a transformation which slightly changes the bitsize of P [2]. Let τ bound the bitsize of the coefficients of P . We wish to find compute a lower bound on m , greater than zero, depending on n, d, τ . Equivalently, we have a system with unknowns m, x_i :

$$\begin{cases} \frac{\partial P}{\partial x_1}(x_1, \dots, x_n) = \dots = \frac{\partial P}{\partial x_n}(x_1, \dots, x_n) = 0, \\ P(x_1, \dots, x_n) = m. \end{cases} \quad (21)$$

We use Th. 11, since there is no guarantee that the system is 0-dimensional. However, Th. 11 provides bounds for the isolated points of the variety. Since the minimum could be attained on a non-zero dimensional component, we should argue that the bounds take care of this case. We consider all the irreducible components of the variety defined by (21). Each of them contains a point for which the bounds of Th. 11 apply. Such a point is the limit of a solution of the perturbed system depending on the parameter s when $s \rightarrow 0$. Moreover, it is a zero of the first non-zero coefficient T_U , seen as a polynomial in s [8, 10]; Th. 11 bounds these zeros. Now, on each of these components, the value of m is constant, since

the gradient of P is 0, and so the bounds apply for it as well.

Let $P_i = \frac{\partial P}{\partial x_i}$ and $P_{n+1} = P - m$. It holds that $\deg(P_{n+1}) = d$, $\deg(P_i) \leq d - 1$, $\|P_{n+1}\|_\infty \leq 2^\tau$, $\|P_i\|_\infty \leq d\|f_{n+1}\|_\infty \leq d 2^\tau$, $M_{n+1} \leq (d - 1)^n$, $M_i \leq d(d - 1)^{n-1}$, and $D \leq M_0 \leq d(d - 1)^n$. Using (20) we deduce $1/m \leq 2^D \varrho C A$. It remains to bound the various quantities involved, defined in (4):

$$\begin{aligned} C &\leq \prod_{i=1}^{n+1} \|P_i\|_\infty^{M_i} = \|P_{n+1}\|_\infty^{M_{n+1}} \prod_{i=1}^n \|P_i\|_\infty^{M_i} \\ &\leq (2^\tau)^{(d-1)^n} \prod_{i=1}^n (d 2^\tau)^{d(d-1)^{n-1}} \leq 2^{\tau(d-1)^n} (d 2^\tau)^{nd(d-1)^{n-1}} \\ &\leq 2^{(n+1)\tau d(d-1)^{n-1} + nd(d-1)^{n-1} \lg d}, \end{aligned}$$

$$\begin{aligned} A &= \prod_{i=1}^{n+1} \sqrt{M_i} 2^{M_i} = \sqrt{M_{n+1}} \cdot 2^{M_{n+1}} \cdot \prod_{i=1}^n \sqrt{M_i} \cdot 2^{M_i} \\ &\leq (d - 1)^{n/2} \cdot 2^{(d-1)^n} \cdot d^{n/2} (d - 1)^{n(n-1)/2} \cdot 2^{nd(d-1)^{n-1}} \\ &\leq 2^{(n+1)d(d-1)^{n-1} + (n^2+n) \lg \sqrt{d}}. \end{aligned}$$

Moreover, $(\#Q_{n+1}) \leq 2d^{n+1}$, $(\#Q_i) \leq 2(d - 1)^{n+1}$, and so

$$\begin{aligned} \varrho &= \prod_{i=1}^{n+1} (\#Q_i)^{M_i} = (\#Q_{n+1})^{M_{n+1}} \prod_{i=1}^n (\#Q_i)^{M_i} \\ &\leq (2d^{n+1})^{(d-1)^n} \cdot \prod_{i=1}^n (2d^n)^{d(d-1)^{n-1}} \leq 2^{(n+1)(1+(n+1) \lg d)d(d-1)^{n-1}} \end{aligned}$$

We apply (10) using the previous inequalities, and get

$$\frac{1}{m} \leq 2^{(n^2+n) \lg \sqrt{d} + (1+2n+d+(n^2+3n+1) \lg d)d(d-1)^{n-1}} \times 2^{(n+1)\tau d(d-1)^{n-1}}.$$

To assure that the minimum is attained inside the simplex, we apply a transformation that preserves the degree, but the bitsize of the polynomial is now bounded by $\tau + 1 + d \lg(n)$. Replacing this in the previous inequality, we get $\frac{1}{m} \leq \frac{1}{m_{\text{DMM}_P}}$, where

$$\frac{1}{m_{\text{DMM}_P}} = 2^{(n^2+n) \lg \sqrt{d} + (2+3n+d+(n^2+3n+1) \lg d) \cdot 2^{(n+1)d \lg n} d(d-1)^{n-1}} \times 2^{(n+1)\tau d(d-1)^{n-1}}. \quad (22)$$

If we know that the system is zero dimensional then we could use Th. 3. Of course this is not always the case, hence we state the following bound, using (5), just as a reference.

$$\frac{1}{m} \leq \frac{1}{m_{\text{DMM}}} = 2^{((n+1)\tau + n + d + (n^2+3n+1) \lg d)d(d-1)^{n-1}}. \quad (23)$$

Let us compare the m_{DMM_P} with other bounds that appear in the bibliography. In [2, Sec. 2, Rem. 2.17], the following estimation was computed,

$$\begin{aligned} \frac{1}{m_{\text{BLR}}} &= 2^{2^{n+3} n d^{n+1} (\tau + 8nd)} n^{2^{n+5} d^{n+2} n} d^{2^{n+5} d^{n+1} n^2} \\ &= 2^{2^{n+3} n \tau d^{n+1} + 2^{n+5} n d^{n+1} (2nd + d \lg n + n \lg d)}. \end{aligned} \quad (24)$$

which also holds with no assumption, but it is looser than m_{DMM_P} .

In [5] the authors derive a bound for the minimum of the absolute value of a polynomial, $\frac{1}{m} \leq \frac{1}{m_{\text{BY}}}$, i.e.

$$\frac{1}{m_{\text{BY}}} = ((n+2)^2 e^{n+3})^{(n+1)(n+2)d^{n+1}} (n^n (n+1) d 2^\tau)^{(n+1)d^n}. \quad (25)$$

The authors use the terminology *evaluation bound* for their bound. It holds when there is a 0-dimensional projection; they prove that this is always the case for (21).

In [16] the following bound was computed:

$$\frac{1}{m} \leq \frac{1}{m_{\text{JP}}} = 2^{(\tau+1)d^{n+1}} d^{(n+1)d^{n+1}}, \quad (26)$$

which has no restriction on the corresponding polynomial system. It is comparable to m_{DMM_p} in general, but strictly looser when $d > n$.

Example 12. Let us compute a lower bound on the value of $f = (x+2y-3)^d + (x+2y-4)^d$, where $d \in \{2, 8, 32\}$. The polynomial is positive as it is a sum of squares. Consider the ideal $I = (f - z, f_x, f_y) \subset \mathbb{Z}[x, y, z]$. If $(\zeta, \zeta_2, \zeta_3)$ belongs to the zero set of I_f , then $|\zeta_3| \geq 2^{-b}$, $b > 0$. In Tab. 1 we present the estimations of $\lg b$ by the previous bounds. The true value is $b = 0$. When the degree is comparable to the number of variables ($d = 2$), then our bound and m_{JP} are comparable. When $d > n$, e.g. $d = 4$ and $d = 32$, then m_{DMM_p} is better than m_{JP} by an order of magnitude.

5. SUBDIVISION ALGORITHMS

We use our results to bound the number of steps that any subdivision algorithm performs to isolate the real roots of a well-defined polynomial system. Then, we bound the complexity of Milne's algorithm in 2d. Our analysis can easily be extended to $\mathbb{R}^n$, however it is not clear what is the exact bit complexity of the elimination steps needed.

We use DMM_n , Th. 3, and Eq. (4) & (9), to bound the number of steps of a subdivision algorithm to isolate the real roots of a well-defined polynomial system as in (1). We assume the existence of an oracle that counts the number of real roots of the system inside a box in $\mathbb{Q}^n$. Our aim is to compute the number of calls to the oracle in order to compute isolating (hyper-)boxes for all real roots. Realizations of such oracles are in [22, 25, 24], see also [3].

Suppose all roots of the system lie in a hypercube of side C , see Th. 3. At step h of the algorithm, the oracle counts the number of roots in hypercubes of side $C/2^h$. We consider the whole subdivision algorithm as a 2^n -ary tree T , where at each node we associate a hypercube, and to the root of the tree we associate the initial hypercube. Let $\#(T)$ denote the number of nodes. We will prune some leaves of T to obtain tree T' where it is easier to count its nodes.

We proceed as follows. If v is a leaf and has a sibling that it is not a leaf, then we prune v . If $u_1, \dots, u_k$, for some positive integer k , are leaves and siblings, such that they have no sibling that is not a leaf, then we prune all of them except one that possess a hypercube that contains a real root. Notice that there is always at least one such node in $u_1, \dots, u_k$, because otherwise, the subdivision process in this path would have stopped one level before. If there exists more than one such node in $u_1, \dots, u_k$, then we keep arbitrarily one of them. It holds that $\#(T) \leq 2^n \#(T')$, and we will count the nodes in T' .

Each leaf of the tree contains a hypercube that isolates a real root of the system, and if there are at most R real roots, this also bounds the number of the leaves of T' . The hypercubes that correspond to the leaves of the tree have diagonals that are at least $\Delta_j = |\gamma_j - \gamma_{c_j}|$, where γ_{c_j} is the root closest to γ_j . The length of their edges is at least $|\gamma_{j,i} - \gamma_{c_j,i}|$, where $1 \leq i \leq n$. It holds that $\Delta_j = |\gamma_j - \gamma_{c_j}| \geq |\gamma_{j,i} - \gamma_{c_j,i}|$, for any index i . The number of nodes from a leaf to the root of the tree is $\left\lceil \log \frac{C}{\Delta_j} \right\rceil$. Hence the number

of nodes in $\#(T')$ is

$$\#(T') = \sum_{j=1}^R \left\lceil \log \frac{C}{\Delta_j} \right\rceil \leq R + R \lg C - \lg \prod_{j=1}^R \Delta_j. \quad (27)$$

To bound the various quantities that appear, we will rely on Eq. (4) and Th. 3. If the total degree of the polynomials is bounded by d , and $\|f_i\|_\infty \leq 2^\tau$, then $\lg C \leq n \tau d^{n-1}$. To bound $\prod_{j=1}^R \Delta_j$ we use Eq. (4) of Th. 3 with $\ell = R$. The hypotheses of the theorem, concerning the indices of the roots, are not fulfilled when symmetric products occur. In this case, we factorize quantity as $\prod_{i=1}^R \Delta_i = \prod_{i=1}^{R_1} \Delta_i \prod_{i=1}^{R_2} \Delta_i$, where $R_1 + R_2 = R$ and the factors are such that no symmetric products occur. Then

$$\prod_{i=1}^R \Delta_i = \prod_{i=1}^{R_1} \Delta_i \prod_{i=1}^{R_2} \Delta_i \geq 2^{-R-(D-1)(D+2)} (hC)^{2-2D-R} B^{-(n-1)(2D^2+D(R+2)+R)}.$$

If we take into account that $R \leq D \leq d^n$, then

$$-\log \prod_{i=1}^R \Delta_i \leq 2D^2 + 3D \lg C + 3D \lg h + 5n D^2 \lg B \leq 8(\lg n + n \lg d) d^{2n} + 3n(n \lg d + \tau) d^{2n-1},$$

and for the total number of nodes of T' we have

$$\begin{aligned} (\#T') &\leq R + R \lg C - \lg \prod_{j=1}^R \Delta_j \leq D + D \lg C - \lg \prod_{j=1}^R \Delta_j \\ &\leq 2d^n (n \tau d^{n-1}) + 8(\lg n + n \lg d) d^{2n} + 3n(n \lg d + \tau) d^{2n-1} \\ &= \tilde{O}(n(n + d + \tau) d^{2n-1}), \end{aligned}$$

and hence $(\#T) = \tilde{O}(2^n n(n + d + \tau) d^{2n-1})$.

Theorem 13. Consider the polynomial system formed by the polynomials in (1). The number of steps that a subdivision algorithm performs in order to compute isolating boxes for all the real roots of the system is $\tilde{O}(2^n D(D + \lg C))$ or $\tilde{O}(2^n (d + \tau) d^{2n-1})$.

Remark 14. If we specialize $n = 1$ in the previous theorem, then we deduce that the number of steps of subdivisions algorithms for real root isolation of univariate integer, not necessarily square-free, polynomials is $\mathcal{O}(d^2 \lg d + d\tau)$. The optimal bound is $\mathcal{O}(d^2 + d\tau)$ [11].

We now bound the complexity of Milne's algorithm [22] for isolating all real roots of a bivariate polynomial system. Milne's, so-called, *volume function* realizes the required oracle, see [15, 32] for experimental results. By $\mathbf{SR}(f, g)$ we denote the signed polynomial remainder sequence of f, g .

Proposition 15. [26, 12] We compute $\mathbf{SQ}(f, g)$, any polynomial in $\mathbf{SR}(f, g)$, and $\mathbf{Res}(f, g)$ wrt x in $\tilde{O}_B(q(p+q)^{k+1} d \tau)$. The degree of $\mathbf{SR}(f, g)$ in $y_1, \dots, y_k$ is $\mathcal{O}(d(p+q))$ and the bitsize is $\mathcal{O}((p+q)\tau)$. We can evaluate $\mathbf{SR}(f, g)$ at $x = \mathbf{a}$, where $\mathbf{a} \in \mathbb{Q} \cup \{\infty\}$ and $\mathcal{L}(\mathbf{a}) = \sigma$, in $\tilde{O}_B(q(p+q)^{k+1} d \max\{\tau, \sigma\})$.

Let $f, g \in \mathbb{Z}[x, y]$ with total degrees bounded by d and bitsize bounded by τ . We are interested in isolating the real roots of the polynomial system $f(x, y) = g(x, y) = 0$, which we assume to be 0-dimensional. We introduce new parameters u, a, b and we eliminate a, b from the polynomials $\{f(a, b), g(a, b), V = u + (x - a)(y - b)\}$, where V is the volume function. After elimination, we obtain a polynomial $h \in (\mathbb{Z}[x, y])[u]$. We compute the Sturm sequence of h and its derivative w.r.t. u , h_u , and we evaluate the sequence over $u = 0$. We obtain a sequence of bivariate polynomials in x, y . Now consider a box in the plane. We evaluate the sequence on each vertex of the box, and we count the number

bound		$(d, \tau) = (2, 5)$	$(8, 20)$	$(32, 85)$
[2], Eq. (24)	$ \lg(m_{BLR}) $	27 136	6 684 672	1 604 321 280
[5], Eq. (25)	$ \lg(m_{BY}) $	1 192	74 000	4 696 811
[16], Eq. (26)	$ \lg(m_{JP}) $	72	15 360	3 309 568
Eq. (22)	$ \lg(m_{DMM_p}) $	87	7 457	442 447
Eq. (23)	$ \lg(m_{DMM}) $	54	5 201	324 506

Table 1. Comparison of (the bitsize of) various bounds on the minimum value of the polynomial $f = (x + 2y - 3)^d + (x + 2y - 4)^d$, for $d \in \{2, 8, 32\}$ and $\tau \in \{8, 20, 85\}$, resp. The bounds hold for all polynomials with same characteristics.

of sign variations. The number of real roots inside the box is $\frac{1}{4}$ the sum of the sign variations [22].

We perform the elimination using iterated resultants. Using Prop. 15 we compute $h_1 = \text{Res}_a(f(a, b), V(u, x, y, a, b)) \in \mathbb{Z}[u, x, y, a, b]$ in $\tilde{\mathcal{O}}_B(d^7\tau)$. The total degree of h_1 is $\mathcal{O}(d^2)$ and $\mathcal{L}(h_1) = \tilde{\mathcal{O}}(d\tau)$. Similarly, we obtain polynomial $h_2 = \text{Res}_a(g(a, b), V(u, x, y, a, b)) \in \mathbb{Z}[u, x, y, a, b]$. Finally, $h = \text{Res}_b(h_1, h_2) \in \mathbb{Z}[x, y, u]$ is computed in $\tilde{\mathcal{O}}_B(d^{12}\tau)$. The degree of h in u is $\mathcal{O}(d^2)$ since the resultant of h_1, h_2 has the factor $u^{\deg(f(x,0))\deg(g(x,0))} = u^{d^2}$. The degree of h in x, y is $\tilde{\mathcal{O}}_B(d^4)$ and $\mathcal{L}(h) = \tilde{\mathcal{O}}(d^3\tau)$.

We compute the signed polynomial remainder sequence of h, h_u and evaluate it at 0. This costs $\tilde{\mathcal{O}}_B(d^{15}\tau)$. The evaluated sequence contains $\mathcal{O}(d^2)$ polynomials in $\mathbb{Z}[x, y]$ of degrees $\tilde{\mathcal{O}}_B(d^6)$ and bitsize $\tilde{\mathcal{O}}_B(d^5\tau)$. Each polynomial in the sequence is evaluated over a rational number of bitsize σ in $\tilde{\mathcal{O}}_B(d^{17}(\tau + d\sigma))$, and thus all of them in $\tilde{\mathcal{O}}_B(d^{19}(\tau + d\sigma))$.

In the worst case, σ equals the bitsize of the separation bound, i.e. $\tilde{\mathcal{O}}(d^3\tau)$. Hence, the evaluation of the sequence costs $\tilde{\mathcal{O}}_B(d^{23}\tau)$. Th. 13 indicates that we need to perform this evaluation $\mathcal{O}(d^4 \lg d + d^3\tau)$ times.

Theorem 16. *Let $f, g \in \mathbb{Z}[x, y]$ with total degrees bounded by d and bitsize bounded by τ . Using the algorithm of Milne [22], we can isolate the real roots of the system $f = g = 0$ in $\tilde{\mathcal{O}}_B(d^{27}\tau + d^{26}\tau^2)$.*

Bounds on mutli-point evaluation of multivariate polynomials [23] could save at least two factors in the previous theorem.

Acknowledgment. E.T. thanks M. Sombra for finding a missing factor in the original manuscript, and brought to our attention [28]. IZE and BM are partially supported by Marie-Curie Network “SAGA”, FP7 contract PITN-GA-2008-214584. ET is partially supported by an individual postdoctoral grant from the Danish Agency for Science, Technology and Innovation.

6. REFERENCES

- [1] E.H. Bareiss. Sylvester’s identity and multistep integer-preserving Gaussian elimination. *Math. of Comput.*, 22(103):565–578, 1968.
- [2] S. Basu, R. Leroy, and M-F. Roy. A bound on the minimum of the real positive polynomial over the standard simplex. Technical Report arXiv:0902.3304v1, arXiv, Feb 2009.
- [3] S. Basu, R. Pollack, and M-F. Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms & Comput. in Math.* Springer-Verlag, 2nd edition, 2006.
- [4] H. F. Blichfeldt. A new principle in the geometry of numbers, with some applications. *Trans. AMS*, 15(3):227–235, 1914.
- [5] W. D. Brownawell and C. K. Yap. Lower bounds for zero-dimensional projections. In *Proc. ISSAC*, KIAS, Seoul, Korea, 2009.
- [6] M. Burr, S.W. Choi, B. Galehouse, and C. K. Yap. Complete subdivision algorithms, II: Isotopic meshing of singular algebraic curves. In *Proc. ISSAC*, pages 87–94, Hagenberg, Austria, 2008.
- [7] J. Canny. *The Complexity of Robot Motion Planning*. ACM Doctoral Dissertation Award Series. MIT Press, 1987.
- [8] J. Canny. Generalised characteristic polynomials. *J. Symbolic Computation*, 9(3):241–250, 1990.
- [9] D. Cox, J. Little, and D. O’Shea. *Using Algebraic Geometry*. Number 185 in GTM. Springer, New York, 2nd edition, 2005.
- [10] C. D’Andrea and I.Z. Emiris. Computing sparse projection operators. *Contemporary Mathematics*, 286:121–140, 2001.
- [11] J. H. Davenport. Cylindrical algebraic decomposition. Technical Report 88–10, School of Math. Sciences, Univ. Bath, <http://www.bath.ac.uk/masjhd/>, 1988.
- [12] D. I. Diochnos, I. Z. Emiris, and E. P. Tsigaridas. On the asymptotic and practical complexity of solving bivariate systems over the reals. *J. Symb. Comput.*, 44(7):818–835, 2009.
- [13] A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the Descartes method. In *Proc. ISSAC*, pages 71–78, New York, USA, 2006.
- [14] I. Z. Emiris. *Sparse Elimination and Applications in Kinematics*. PhD thesis, Computer Science Division, Univ. of California at Berkeley, December 1994.
- [15] L. González-Vega and G. Trujillo. Multivariate Sturm-Habicht sequences: Real root counting on n-rectangles and triangles. *Real Algebraic and Analytic Geometry (Segovia, 1995)*, *Rev. Mat. Univ. Complut. Madrid*, 10:119–130, 1997.
- [16] G. Jeronimo and D. Perrucci. On the minimum of a positive polynomial over the standard simplex. *CoRR*, abs/0906.4377, 2009.
- [17] J. R. Johnson. *Algorithms for Polynomial Real Root Isolation*. PhD thesis, The Ohio State Univ., 1991.
- [18] T. Krick, L.M. Pardo, and M. Sombra. Sharp estimates for the arithmetic Nullstellensatz. *Duke Mathematical Journal*, 109(3):521–598, 2001.
- [19] A. Mantzaflaris, B. Mourrain, and E.P. Tsigaridas. Continued fraction expansion of real roots of polynomial systems. In *Proc. Symbolic-Numeric Comput.*, pages 85–94, Kyoto, 2009.
- [20] M. Mignotte. *Mathematics for computer algebra*. Springer-Verlag, New York, 1991.
- [21] M. Mignotte. On the Distance Between the Roots of a Polynomial. *Appl. Algebra Eng. Commun. Comput.*, 6(6):327–332, 1995.
- [22] P. S. Milne. On the solution of a set of polynomial equations. In B. Donald, D. Kapur, and J. Mundy, editors, *Symbolic & Numerical Computation for AI*, pages 89–102. 1992.
- [23] M. Nüsken and M. Ziegler. Fast multipoint evaluation of bivariate polynomials. In S. Albers and T. Radzik, editors, *ESA*, volume 3221 of *Lecture Notes in Computer Science*, pages 544–555. Springer, 2004.
- [24] P. Pedersen. *Counting real zeros*. PhD thesis, NY Univ., 1991.
- [25] P. Pedersen, M-F. Roy, and A. Szpirglas. Counting real zeros in the multivariate case. In F. Eyssette and A. Galligo, editors, *Computational Algebraic Geometry*, volume 109 of *Progress in Mathematics*, pages 203–224. Birkhäuser, Boston, 1993.
- [26] D. Reischert. Asymptotically fast computation of subresultants. In *Proc. ISSAC*, pages 233–240, 1997.
- [27] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *J. of Appl. Algebra in Engin., Comm. and Computing*, 9(5):433–461, 1999.
- [28] M. Sombra. The height of the mixed sparse resultant. *Amer. J. Math.*, 126:1253–1260, 2004.
- [29] Elias P. Tsigaridas and Ioannis Z. Emiris. On the complexity of real root isolation using Continued Fractions. *Theor. Comput. Sci.*, 392:158–173, 2008.
- [30] J-C. Yakoubsohn. Numerical analysis of a bisection-exclusion method to find zeros of univariate analytic functions. *J. Complexity*, 21(5):652–690, 2005.
- [31] C. K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.
- [32] Z. Zafeirakopoulos. Study and benchmarks for real root isolation methods. Master’s thesis, Dept. Informatics & Telecoms, University of Athens, 2009. www.zafeirakopoulos.info/content/publications/thesis.pdf.

Solving Bezout-Like Polynomial Equations for the Design of Interpolatory Subdivision Schemes

C. Conti
Dipartimento di Energetica,
Università di Firenze
Via Lombroso 6/17, 50134,
Firenze, Italy
costanza.conti@unifi.it

L. Gemignani
Dipartimento di Matematica,
Università di Pisa
Largo B. Pontecorvo, 56127,
Pisa, Italy
gemignan@dm.unipi.it

L. Romani
Dipartimento di Matematica
e Applicazioni,
Università di Milano-Bicocca
Via R. Cozzi 53, 20125,
Milano, Italy
lucia.romani@unimib.it

ABSTRACT

Subdivision schemes are nowadays customary in curve and surface modeling. In this paper the problem of designing interpolatory subdivision schemes is considered. The idea is to modify a given approximating subdivision scheme just enough to satisfy the interpolation requirement. From an algebraic point of view this leads to the solution of a generalized Bezout polynomial equation possibly involving more than two polynomials. By exploiting the matrix counterpart of this equation it is shown that small-degree solutions can be generally found by inverting an associated structured matrix of Toeplitz-like form. If the approximating scheme is defined in terms of a free parameter, then the inversion can be performed by numeric-symbolic methods.

Categories and Subject Descriptors

I.3 [Computer Graphics]: Computational Geometry and Object Modeling; G.1 [Numerical Analysis]: Numerical Linear Algebra

General Terms

Algorithms

Keywords

Subdivision scheme, Bezout equation, structured matrix

1. INTRODUCTION

Subdivision schemes are widely used in several contexts like in Computer Aided Geometric Design to represent a smooth curve or surface as the limit of successive refinements on denser and denser grids of points. At each refinement step new points are generated by means of an affine combination of the existing ones. The number of inserted points characterizes the concept of *arity* of the considered scheme. For instance, if the points are doubled at each iteration the scheme

is called binary its arity being $p = 2$. For processes defined in the real two- or three-dimensional Euclidean space the analysis can be immediately reduced to the scalar univariate case by restriction to each coordinate. A *stationary univariate subdivision scheme* with *arity* $p \geq 2$ is an iterative process that starting with some initial points attached to the integer grid, *i.e.*, with $\mathbf{q} = \mathbf{q}^{(0)} = (q_i^{(0)} : i \in \mathbb{Z})$, iteratively computes a sequence $\mathbf{q}^{(n)} = S_{\mathbf{a}}\mathbf{q}^{(n-1)} = S_{\mathbf{a}}^n\mathbf{q}^{(0)}$ for $n \geq 1$, by repeated application of the stationary rule

$$\left(\mathbf{q}^{(n)}\right)_i = \left(S_{\mathbf{a}}\mathbf{q}^{(n-1)}\right)_i = \sum_{j \in \mathbb{Z}} a_{i-pj} q_j^{(n-1)}, \quad i \in \mathbb{Z}, \quad (1)$$

which relies upon the coefficients a_i , $i \in \mathbb{Z}$. These coefficients identify the subdivision operator $S_{\mathbf{a}}$ and the so called *refinement mask* $\mathbf{a} = (a_i : i \in \mathbb{Z})$. It is assumed that $\sigma(\mathbf{a}) = \{j \in \mathbb{Z} : a_j \neq 0\} \subseteq [-N, N]$ for a certain $N \in \mathbb{N}$ so that $\mathbf{a} \in \ell_0(\mathbb{Z})$, *i.e.*, the space of compactly supported sequences of real values. For the purpose of analysis it is also useful to introduce the *symbol*

$$a(z) = \sum_{i=-N}^N a_i z^i, \quad z \in \mathbb{C} \setminus \{0\},$$

which is a Laurent polynomial associated with the mask $\mathbf{a}$, and, moreover, the associated *sub-symbols*

$$a_j(z) = \sum_{i=-N}^N a_{pi+j} z^i, \quad z \in \mathbb{C} \setminus \{0\}, \quad 0 \leq j \leq p-1.$$

These are related to the symbol by the equation

$$a(z) = \sum_{i=0}^{p-1} z^i a_i(z^p).$$

By assigning the values of $S_{\mathbf{a}}^n\mathbf{q}$, $n \in \mathbb{N}_0$, to the denser and denser grids $p^{-n}\mathbb{Z}$, one can then establish a notion of L_{∞} -convergence to a continuous limit function by requiring the existence of a uniformly continuous and bounded function $f_{\mathbf{q}}$ (depending on the starting sequence $\mathbf{q}$) satisfying

$$\lim_{n \rightarrow \infty} \sup_{j \in \mathbb{Z}} \left| (S_{\mathbf{a}}^n\mathbf{q})_j - f_{\mathbf{q}}(p^{-n}j) \right| = 0. \quad (2)$$

We say that the subdivision scheme (1) is convergent if it converges for any initial vector $\mathbf{q} = \mathbf{q}^{(0)} \in \ell_{\infty}(\mathbb{Z})$, *i.e.*, $\|\mathbf{q}\|_{\infty} := \sup_{i \in \mathbb{Z}} |q_i| < \infty$, and, moreover, $f_{\mathbf{q}} \neq 0$ for at least some initial data $\mathbf{q} \in \ell_{\infty}(\mathbb{Z})$.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

An important class of subdivision schemes are those that refine the sequence $\mathbf{q}$ while keeping the “original data” in the sense that $(S_{\mathbf{a}}\mathbf{q})_{pi} = q_i$, $i \in \mathbb{Z}$. For obvious reasons, such schemes are called *interpolatory* and their refinement mask is of special type since it satisfies

$$a_{pi} = \delta_{i,0}, \quad i \in \mathbb{Z}. \quad (3)$$

The condition (3) can also be easily reformulated in terms of the associated symbol: the Laurent polynomial $a(z)$ is the symbol of an interpolatory subdivision scheme of arity p if and only if it satisfies

$$\sum_{j=0}^{p-1} a(r_j z) = p, \quad z \in \mathbb{C} \setminus \{0\}, \quad (4)$$

where $r_j = e^{\frac{2\pi i j}{p}}$, $j = 0, \dots, p-1$ are the p -th roots of the unity. Despite of their importance and of the recent burgeoning literature in the field of subdivision schemes (see for instance [5, 12] and the references therein), very few interpolatory examples are known so far even in the univariate setting. The most celebrated example is the class of Dubuc-Deslauriers (DD) symmetric schemes, first presented in [3].

In this paper we present a general approach for deriving a family of interpolatory subdivision schemes associated with the subdivision operator defined in (1). The approach generalizes previous results in [2, 4, 8, 9]. Since in a matrix setting this (linear) operator can be represented by a bi-infinite banded Toeplitz-like matrix $S_{\mathbf{a}}$ we argue that a novel subdivision operator $S_{\mathbf{b}}$ of the same form as $S_{\mathbf{a}}$ can be obtained by multiplication by a bi-infinite banded Toeplitz matrix $\mathcal{T}$, i.e., $S_{\mathbf{b}} = \mathcal{T} \cdot S_{\mathbf{a}}$. Furthermore, there follows that the symbol $b(z)$ of $S_{\mathbf{b}}$ satisfies $b(z) = a(z) \cdot t(z)$, where $t(z)$ is the symbol of $\mathcal{T}$. In this way we may determine the Laurent polynomial $t(z)$ so that $b(z)$ fulfills the interpolation requirement (4), i.e.,

$$a(z) \cdot t(z) + a(r_1 z) \cdot t(r_1 z) + \dots + a(r_{p-1} z) \cdot t(r_{p-1} z) = p. \quad (5)$$

The solution of such a polynomial equation is at the core of our method for the construction of interpolatory subdivision schemes. By defining the shifted polynomials $\hat{a}(z) = z^N a(z)$, with $\kappa = \deg(\hat{a}(z))$, and $\hat{t}(z) = z^h t(z)$, with $m = \deg(\hat{t}(z)) < \kappa$, where we assume $0 \leq N + h \leq \kappa + m$ with $N + h \equiv k \pmod{p}$, $1 \leq k \leq p-1$, it is found that (5) is a specific instance of the customary Bezout-like equation

$$\sum_{j=0}^{p-1} \hat{a}(r_j z) \cdot \hat{f}^{(j)}(z) = p z^{N+h},$$

where we seek polynomials $\hat{f}^{(j)}(z)$, $0 \leq j \leq p-1$, of degree at most $m \leq \kappa - 1$ such that $\hat{f}^{(j)}(z) = r_j^{p-k} \hat{f}^{(0)}(r_j z)$. The shifted polynomial $t(z) = \hat{f}^{(0)}(z) z^{-h}$ provides a solution to (5). The solvability of the general Bezout-like equation for more than two polynomials is studied in [1, 6, 15]. To take advantage of the special form of the polynomials here we nicely apply linear algebra techniques. In a matrix setting the solution of (5) reduces to solving a complex Sylvester-like linear system. Under mild assumptions it is shown that the coefficient matrix can be decomposed in terms of smaller real structured matrices of Hurwitz type. The decomposition affords an effective means for solving (5) as well as for investigating the existence of small degree solutions under given assumptions.

The paper is organized as follows. In Section 2 we present the derivation of the polynomial equation (5) and develop solution methods using structured matrix computations. Some computational examples are illustrated in Section 3. A brief discussion of the results and possible further developments are finally drawn in Section 4.

2. THE POLYNOMIAL APPROACH AND ITS MATRIX COUNTERPARTS FOR GENERATING INTERPOLATORY SUBDIVISION SCHEMES

In the matrix environment the linear operator $S_{\mathbf{a}}$ defined in (1) is represented by a bi-infinite Toeplitz-like matrix $S_{\mathbf{a}} = (a_{i-pj})$, $i, j \in \mathbb{Z}$. Since $a(z) = \sum_{j=-N}^N a_j z^j$, $a_{-N} \neq 0$, is a Laurent polynomial it follows that $S_{\mathbf{a}}$ is banded with bandwidth $\lceil N/p \rceil$ at most. Let $t(z) = \sum_{j=-h}^{\tilde{h}} t_j z^j$, $t_{-h} \neq 0$, be another Laurent polynomial and denote by $\mathcal{T}$ the bi-infinite Toeplitz matrix associated with $t(z)$, namely, $\mathcal{T} = (t_{i-j})$. Observe that $\mathcal{T}$ is again banded with bandwidth $\max\{h, \tilde{h}\}$ at most. For the product operator

$$S_{\mathbf{b}} = \mathcal{T} \cdot S_{\mathbf{a}} = (s_{i,j}), \quad i, j \in \mathbb{Z},$$

we have

$$s_{i,j} = \sum_{r=i-\tilde{h}}^{i+h} t_{i-r} a_{r-pj} = \sum_{\ell=-h}^{\tilde{h}} t_{\ell} a_{i-pj-\ell} = s_{i+p,j+1}, \quad i, j \in \mathbb{Z}.$$

This means that the product operator $S_{\mathbf{b}}$ is a bi-infinite Toeplitz-like matrix of the same form as the subdivision operator $S_{\mathbf{a}}$ with entries $s_{i,j} = s_{i-p,j}$, $i, j \in \mathbb{Z}$. By setting

$$b(z) = a(z)t(z) = \sum_{j=-h-N}^{\tilde{h}+N} b_j z^j$$

where

$$b_j = 0 \text{ if } j \notin [-h-N, \tilde{h}+N],$$

then we find that

$$b_j = \sum_{i=-h}^{\tilde{h}} t_i a_{j-i}, \quad -h-N \leq j \leq \tilde{h}+N,$$

and, therefore,

$$b_{i-pj} = s_{i,j} = s_{i-p,j}, \quad i, j \in \mathbb{Z}.$$

The product operator $S_{\mathbf{b}}$ can therefore be seen as the subdivision operator associated with the Laurent polynomial $b(z)$.

The unknown coefficients of $t(z)$ might be determined in such a way that the resulting polynomial $b(z)$ verifies the analogue of condition (3), i.e.,

$$b_{pi} = \delta_{i,0}, \quad i \in \mathbb{Z}. \quad (6)$$

Let

$$b(z) = \sum_{\ell=0}^{p-1} z^{\ell} b_{\ell}(z^p), \quad t(z) = \sum_{\ell=0}^{p-1} z^{\ell} t_{\ell}(z^p) \quad z \in \mathbb{C} \setminus \{0\},$$

be the representations of $b(z)$ and $t(z)$, respectively, in terms of their sub-symbols. The condition (6) is equivalent to

$b_0(z^p) = 1$. In addition, from

$$\sum_{j=0}^{p-1} b(r_j z) = \sum_{j=0}^{p-1} \sum_{\ell=0}^{p-1} r_j^\ell z^\ell b_\ell(z^p) =$$

$$\sum_{\ell=0}^{p-1} z^\ell b_\ell(z^p) \left(\sum_{j=0}^{p-1} r_j^\ell \right) = p \cdot b_0(z^p),$$

where $r_j = e^{\frac{2\pi i j}{p}}$, $j = 0, \dots, p-1$, we conclude that $b(z)$ is the symbol of an interpolatory subdivision scheme of arity p if and only if

$$a(z) \cdot t(z) + a(r_1 z) \cdot t(r_1 z) + \dots + a(r_{p-1} z) \cdot t(r_{p-1} z) = p. \quad (7)$$

The Laurent polynomials $t(z)$ that are solutions of (7) are suited to convert the approximating subdivision scheme associated with $a(z)$ into a corresponding interpolating subdivision scheme associated with $b(z) = a(z) \cdot t(z)$. It is clear that small-degree solutions are particularly interesting as they modify the support of $a(z)$ as little as possible.

In the sequel of this section we investigate conditions under which the (generalized) Bezout equation (7) is solvable and admits small-degree solutions. Moreover, we develop effective computational methods for solving (7) in these interesting cases. First of all, suppose that the shifted symbol $\hat{a}(z) = z^N a(z)$ is a polynomial of degree $\kappa \geq p-1$ and set $m = \lceil \frac{\kappa}{p-1} \rceil - 1$. We look for a Laurent polynomial $t(z)$ solving (7) and determined by $m+1$ nonzero coefficients at most. This polynomial can be represented in the form

$$t(z) = t_{-h} z^{-h} + t_{-h+1} z^{-h+1} + \dots + t_{-h+m} z^{-h+m},$$

where h is determined so that $N+h$ is a certain integer multiple of p , say $N+h = jp$. By multiplying (7) by $z^{N+h} = (r_\ell z)^{N+h}$, $0 \leq \ell \leq p-1$, it follows that $\hat{t}(z) = z^h t(z)$ should satisfy

$$\sum_{\ell=1}^{p-1} \hat{a}(r_\ell z) \cdot \hat{t}(r_\ell z) = p z^{jp}. \quad (8)$$

Since $\deg(\hat{a}(z)) = \kappa$ and $\deg(\hat{t}(z)) = m$ we find that j must satisfy $0 \leq jp \leq \kappa + m$ and, hence, $h = jp - N$ with $0 \leq j \leq m$.

The solution of (8) can be reduced to solving a structured linear system of order

$$\lceil \frac{\kappa}{p-1} \rceil (p-1) + m + 1 = (m+1)p$$

whose coefficient matrix is Sylvester-like. Let

$$\mathbf{a}_0 = [a_{-N}, \dots, a_0, \dots, a_{-N+\kappa}]^T \in \mathbb{R}^{\kappa+1}$$

denote the coefficient vector of the polynomial $\hat{a}(z)$. The associated extended coefficient vector $\hat{\mathbf{a}}_0 \in \mathbb{R}^{(m+1)p}$ is defined by $\hat{\mathbf{a}}_0^T = [\mathbf{a}_0^T, 0, \dots, 0]$. Similarly let us introduce the extended coefficient vectors $\hat{\mathbf{a}}_j \in \mathbb{C}^{(m+1)p}$, $1 \leq j \leq p-1$, associated with the polynomials $\hat{a}(r_j z)$, $1 \leq j \leq p-1$, respectively. Moreover let $Z = (z_{i,j}) \in \mathbb{R}^{(m+1)p \times (m+1)p}$ be the down-shift matrix given by $z_{i,j} = \delta_{i-1,j}$, where $\delta_{i,j}$ is the Kronecker delta symbol. Set $\mathcal{R}_j \in \mathbb{C}^{(m+1)p \times (m+1)p}$ the striped Toeplitz matrix

$$\mathcal{R}_j = [\hat{\mathbf{a}}_j | Z \hat{\mathbf{a}}_j | \dots | Z^m \hat{\mathbf{a}}_j], \quad 0 \leq j \leq p-1.$$

The coefficient matrix of the linear system (8) is

$$\mathcal{R} = [\mathcal{R}_0 | \mathcal{R}_1 | \dots | \mathcal{R}_{p-1}] \in \mathbb{C}^{(m+1)p \times (m+1)p}.$$

The linear system can be written as

$$\mathcal{R} \cdot \begin{bmatrix} \hat{\mathbf{t}}_0^T | \hat{\mathbf{t}}_1^T | \dots | \hat{\mathbf{t}}_{p-1}^T \end{bmatrix}^T = p \cdot \mathbf{e}_{jp+1}, \quad (9)$$

where $0 \leq j \leq m$, $\hat{\mathbf{t}}_k \in \mathbb{C}^{m+1}$ is the coefficient vector of $\hat{t}(r_k z)$, $0 \leq k \leq p-1$, and $\mathbf{e}_{jp+1}$ is the column of given index of the identity matrix of order $(m+1)p$. This means that, whenever $\mathcal{R}$ is nonsingular a complete set of solutions of (8) can be determined by inverting the matrix $\mathcal{R}$.

Remark 1. It is worth noting that in the case $p = 2$ we find that $m+1 = \kappa$ and, moreover, the matrix $\mathcal{R}$ reduces to the classical Sylvester resultant matrix which is invertible if and only if $a(z)$ and $a_1(z) = a(-z)$ are relatively prime. This property is exploited in [2] to derive a complete characterization of the solutions of (8). Differently, it is shown below that in general for $p > 2$ the matrix $\mathcal{R}$ is not a resultant matrix as it can be singular even if $\kappa = (m+1)(p-1)$ and the polynomials $a(r_j z)$, $0 \leq j \leq p-1$ have no common factors. Extensions of the classical resultant theorem for more than two polynomials ($p > 2$) are established in [1, 15], but they employ rectangular generalized resultant matrices.

Example 1. Let

$$a(z) = z^{-3} + 2z^{-1} + 1 + 2z + z^3,$$

with $N = 3$, $\kappa = 6$, $p = 3$ and $m = 2$. Although the polynomials $a(z)$, $a(e^{\frac{2\pi i}{3}} z)$ and $a(e^{\frac{4\pi i}{3}} z)$ are pairwise relatively prime, it is easily found that the corresponding 9×9 matrix $\mathcal{R}$ is singular.

A column permutation applied to $\mathcal{R}$ enables the linear system (9) to be rearranged in a more convenient way. Let $\mathcal{P}_R \in \mathbb{R}^{(m+1)p \times (m+1)p}$ be the permutation matrix such that

$$\hat{\mathcal{R}} = \mathcal{R} \cdot \mathcal{P}_R = [\hat{\mathcal{R}}_0 | Z \hat{\mathcal{R}}_0 | \dots | Z^m \hat{\mathcal{R}}_0], \quad (10)$$

where

$$\hat{\mathcal{R}}_0 = [\hat{\mathbf{a}}_0 | \hat{\mathbf{a}}_1 | \dots | \hat{\mathbf{a}}_{p-1}].$$

We find that

$$\mathcal{P}_R^T \cdot \begin{bmatrix} \hat{\mathbf{t}}_0^T | \hat{\mathbf{t}}_1^T | \dots | \hat{\mathbf{t}}_{p-1}^T \end{bmatrix}^T = [t_{-h} \boldsymbol{\omega}_0^T | t_{1-h} \boldsymbol{\omega}_1^T | \dots | t_{m-h} \boldsymbol{\omega}_m^T]^T,$$

where

$$\boldsymbol{\omega}_\ell = \begin{bmatrix} r_0^\ell \\ \vdots \\ r_{p-1}^\ell \end{bmatrix}, \quad 0 \leq \ell \leq m.$$

Now let $\Omega \in \mathbb{C}^{p \times p}$ the Fourier matrix with columns $\boldsymbol{\omega}_\ell$, $0 \leq \ell \leq p-1$, that is,

$$\Omega = \begin{bmatrix} 1 & r_0 & \dots & r_0^{p-1} \\ 1 & r_1 & \dots & r_1^{p-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & r_{p-1} & \dots & r_{p-1}^{p-1} \end{bmatrix}.$$

Furthermore, let $\mathcal{D}_B$ be the block diagonal matrix given by

$$\mathcal{D}_B = \begin{bmatrix} \Omega^{-1} & & & \\ & \mathcal{C} \cdot \Omega^{-1} & & \\ & & \ddots & \\ & & & \mathcal{C}^m \cdot \Omega^{-1} \end{bmatrix},$$

where $C \in \mathbb{R}^{p \times p}$ denotes the generator of the circulant matrix algebra, i.e.,

$$C = \begin{bmatrix} 0 & 1 & & \circ \\ & & \ddots & \\ \circ & & & 1 \\ 1 & 0 & \dots & 0 \end{bmatrix}.$$

Then $\mathcal{D}_B$ is invertible and, moreover,

$$\mathcal{D}_B \begin{bmatrix} t_{-h} \omega_0^T | t_{1-h} \omega_1^T | \dots | t_{m-h} \omega_m^T \end{bmatrix}^T = \begin{bmatrix} t_{-h} \mathbf{e}_1^T | t_{1-h} \mathbf{e}_1^T | \dots | t_{m-h} \mathbf{e}_1^T \end{bmatrix}^T.$$

In the light of (10) the structure of $\widehat{\mathcal{R}} \cdot \mathcal{D}_B^{-1}$ follows from that one of $\widehat{\mathcal{R}}_0 \cdot \Omega$. Observe that

$$\widehat{a}(z) = \widehat{a}_0(z^p) + z \widehat{a}_1(z^p) + \dots + z^{p-1} \widehat{a}_{p-1}(z^p)$$

with

$$\widehat{a}_\ell(z) = \sum_{j=0}^m a_{-N+\ell+jp} z^j, \quad 0 \leq \ell \leq p-1.$$

It holds

$$\begin{aligned} \sum_{s=0}^{p-1} r_s^\ell \widehat{a}(r_s z) &= \sum_{s=0}^{p-1} \sum_{h=0}^{p-1} r_s^\ell r_s^h z^h \widehat{a}_h(z^p) = \\ \sum_{h=0}^{p-1} \left(\sum_{s=0}^{p-1} r_s^{\ell+h} \right) z^h \widehat{a}_h(z^p) &= p z^j \widehat{a}_j(z^p), \end{aligned}$$

for $j = j(\ell)$ satisfying $j + \ell \equiv 0 \pmod{p}$. There follows that

$$\widehat{\mathcal{R}}_0 \cdot \Omega = p \widehat{\mathcal{R}}_0 = p [\widehat{\mathbf{a}}_0 | Z^{p-1} \widehat{\mathbf{a}}_{p-1} | \dots | Z \widehat{\mathbf{a}}_1],$$

where $\widehat{\mathbf{a}}_\ell$, $0 \leq \ell \leq p-1$, is the extended coefficient vector associated with $\widehat{a}_\ell(z^p)$.

Summing up the linear system (9) can be rewritten in the following form

$$[\widehat{\mathcal{R}}_0 | \dots | Z^m \widehat{\mathcal{R}}_0 \mathcal{C}^m]^T \begin{bmatrix} t_{-h} \mathbf{e}_1^T | \dots | t_{m-h} \mathbf{e}_1^T \end{bmatrix}^T = \mathbf{e}_{jp+1}. \quad (11)$$

It is straightforward to show that the linear system can be reduced to an equivalent system of smaller order. We find that

$$[\widehat{\mathbf{a}}_0 | Z^p \widehat{\mathbf{a}}_{p-1} | Z^p \widehat{\mathbf{a}}_{p-2} | \dots] \begin{bmatrix} t_{-h} \\ t_{1-h} \\ \vdots \\ t_{m-h} \end{bmatrix} = \mathbf{e}_{jp+1}.$$

By performing a suitable rearrangement of rows the system can be finally rewritten in the following form

$$\mathcal{H}_0 \begin{bmatrix} t_{-h} \\ t_{1-h} \\ \vdots \\ t_{m-h} \end{bmatrix} = \mathbf{e}_{j+1}, \quad 0 \leq j \leq m,$$

where $\mathcal{H}_k^T \in \mathbb{R}^{(m+1) \times (m+1)}$, $0 \leq k \leq p-1$, is the finite matrix of Hurwitz type defined by

$$\mathcal{H}_k^T = (h_{i,j}^{(k)}), \quad h_{i,j}^{(k)} = a_{N-k-i+1+(j-1)p}, \quad 1 \leq i, j \leq m. \quad (12)$$

In this way we arrive at the following result.

THEOREM 2.1. *Let us assume that the coefficient matrix $\mathcal{R}$ of (9) is nonsingular. Then $\mathcal{H}_0$ is also nonsingular and, moreover, the complete set of the solutions of (8) is determined by the coefficients of the rows of the inverse of $\mathcal{H}_0^T$.*

Remark 2. The previous theorem describes a subset of the eligible solutions of (7) obtained by first reducing (7) to (8) and then solving this latter polynomial equation by the inversion of $\mathcal{H}_0$. Other different subsets can be found by considering diverse variations of (7). If h is determined so that $N+h \equiv k \pmod{p}$, $1 \leq k \leq p-1$, say $N+h = k+jp$, then by multiplying (7) by $z^{N+h} = (r_\ell z)^{N+h} r_\ell^{p-k}$, $0 \leq \ell \leq p-1$, it follows that the shifted polynomial $\widehat{t}(z) = z^h t(z)$ should satisfy the following modified Bezout-like equation

$$\sum_{\ell=0}^{p-1} r_\ell^{p-k} \widehat{a}(r_\ell z) \widehat{t}(r_\ell z) = p z^{jp+k},$$

where from $0 \leq jp+k \leq \kappa+m$ we get the condition $0 \leq j \leq m$. The modified equation can be solved by using the same approach pursued for (8) thus reducing the computation to invert the finite matrices $\mathcal{H}_k$, $1 \leq k \leq p-1$. Observe that these matrices are finite sections of order $m+1$ of the infinite operator $\mathcal{S}_a$. In this way we establish an intimate connection among the construction of the modified operator $\mathcal{S}_b$, the inversion of the matrices $\mathcal{H}_k$ and their polynomial analogues.

3. COMPUTATIONAL EXAMPLES

In this section we discuss two computational examples related to the application of Theorem 2.1.

Example 2. Following [14], the generalization of binary B-splines of order $m+1$ to arity $p > 2$ is given by the basic limit function associated with the symbol

$$a(z) = \frac{z^{-N}}{p^m} \left(\frac{z^p - 1}{z - 1} \right)^{m+1},$$

with $2N = (p-1) \cdot (m+1)$. We consider the ternary cubic B-spline associated with the approximating subdivision mask (see [7])

$$\mathbf{a} := \frac{1}{27} (\dots, 0, 1, 4, 10, 16, 19, 16, 10, 4, 1, 0, \dots),$$

and we apply the strategy discussed above to get a complete family of corresponding interpolatory ternary subdivision schemes. We have

$$a(z) = \frac{z^4 + 4z^3 + 10z^2 + 16z + 19 + 16z + 10z^2 + 4z^3 + z^4}{27}$$

and, hence, $N = 4$, $p = 3$ and $m = 3$. For $k = 2$ we find that

$$\mathcal{H}_2^T = \frac{1}{27} \begin{bmatrix} 10 & 16 & 1 & 0 \\ 4 & 19 & 4 & 0 \\ 1 & 16 & 10 & 0 \\ 0 & 10 & 16 & 1 \end{bmatrix}$$

whose inverse is

$$\mathcal{H}_2^{-T} = \frac{1}{3} \begin{bmatrix} 14 & -16 & 5 & 0 \\ -4 & 11 & -4 & 0 \\ 5 & -16 & 14 & 0 \\ -40 & 146 & -184 & 27 \end{bmatrix}.$$

By using the coefficients in the second row we define the Laurent correction

$$t(z) = \frac{1}{3} (-4z^{-1} + 11 - 4z)$$

and the “corrected” interpolatory symbol $b(z) = a(z) \cdot t(z)$ with associated mask

$$\mathbf{b} = \frac{1}{81} (\cdots, 0, -4, -5, 0, 30, 60, 81, 60, 30, 0, -5, -4, 0, \cdots).$$

Thus we observe that we can derive the Dubuc-Deslauriers ternary 4-point scheme $(\mathbf{b})$ exactly from the ternary cubic B-spline $(\mathbf{a})$.

We conclude by noticing that for $m+1 = 3, 4, 5, 6, \dots$ many other $(m+1)$ -point schemes of general arity p can be generated in a similar way. In fact, by using the decomposition

$$(1-z)^{-(m+1)} = \sum_{\ell=0}^{\infty} \binom{m+\ell}{m} z^{\ell},$$

after multiplication of the latter one by $(1-z^p)^{m+1}$ we can compute the coefficients $a_i = a_i(p)$ of the corresponding mask and then generate the Laurent correction by symbolic inversion of the Hurwitz-type matrix of order $m+1$.

These schemes include the ones derived in [10, 11] for $m+1 = 3, 4, 5, 6$ by using some specific techniques.

Example 3. Since subdivision schemes are required to provide efficient numerical algorithms to be used in geometric modeling for the design of smooth curves and surfaces, they are frequently defined in terms of one free parameter that the user can suitably set to intuitively modify the final shape. Moreover, due to the fact that subdivision schemes of arity higher than two have the advantage of generating basic limit functions with higher smoothness and smaller support, much attention is deserved to general techniques able to provide arity- p ($p > 2$) interpolatory subdivision schemes with free parameters. It is clear that in all these cases symbolic and computer algebra methods are required to solve the associated Hurwitz-like linear systems by providing an effective way to determine the corresponding interpolating schemes. For the sake of illustration, let us consider the quaternary approximating subdivision scheme introduced in [13] and defined by the symbol

$$\begin{aligned} a(z) &= a_4 z^{-8} + a_8 z^{-7} + a_5 z^{-6} + a_1 z^{-5} + a_3 z^{-4} \\ &+ a_7 z^{-3} + a_6 z^{-2} + a_2 z^{-1} + a_2 + a_6 z + a_7 z^2 \\ &+ a_3 z^3 + a_1 z^4 + a_5 z^5 + a_8 z^6 + a_4 z^7, \end{aligned}$$

where

$$\begin{aligned} a_1 &= (7/32) - (7/64)w, & a_2 &= (29/64) + (13/64)w, \\ a_3 &= (5/16) - (5/64)w, & a_4 &= (1/64) - (1/64)w, \\ a_5 &= (15/128) - (5/64)w, & a_6 &= (57/128) + (7/64)w, \\ a_7 &= (49/128) + (1/64)w, & a_8 &= (7/128) - (3/64)w \end{aligned}$$

and w is a real free parameter. Since $\kappa = 15 = 5 \cdot 3$, we can determine a polynomial correction $\hat{t}(z)$ of degree at most 4 by inverting a certain Hurwitz-type matrix of order 5 with entries depending on the parameter w . It turns out that the 20×20 matrix $\mathcal{R}$ is singular iff $w = 1$ or $w = 1/2$ whereas the 5×5 matrix $\mathcal{H}_2$ is singular for $w = 1/2$. By computing the inverse matrix symbolically we can form the corresponding corrections $\hat{t}(z) = z^h t(z)$. From the entries in the third column of the inverse we get

$$\begin{aligned} t(z) &= \frac{499-594w+176w^2}{-8+16w} + \frac{87-108w+32w^2}{(-8+16w)z^2} + \\ &+ \frac{329-386w+112w^2}{(8-16w)z} + \frac{(371-486w+144w^2)z}{8-16w} + \frac{(53-77w+24w^2)z^2}{-4+8w}. \end{aligned}$$

It is straightforward to verify that $t(z)$ satisfies equation (7) with $p = 4$.

4. CONCLUSIONS AND FUTURE WORK

The examples in Section 3 show that Theorem 2.1 yields an effective way to construct a large family of interpolatory subdivision schemes starting from an initial approximating scheme. The construction essentially reduces to invert some small structured matrices of Hurwitz type defined as finite sections of appropriate order of the linear operator $\mathcal{S}_{\mathbf{a}}$ associated with the approximating scheme. The invertibility of these matrices can be characterized in terms of the invertibility of a corresponding cumulative Sylvester-like matrix. This latter formulation enables the inversion problems for the Hurwitz matrices to be recast into a polynomial setting thus leading to a generalized Bezout-like equation involving $p \geq 2$ polynomials.

In the case $p = 2$ addressed in [2], the Sylvester-like matrix is a resultant matrix and, therefore, the polynomial and the matrix formulations are completely equivalent making possible to translate both the invertibility analysis and the inversion methods into a polynomial framework. In particular, the assumption made in [2] that $a(z)$ is a symmetric Hurwitz polynomial implies that $\mathcal{R}$ is nonsingular and, moreover, the corresponding generalized Bezout equation can be solved by some adaptations of the extended Euclidean algorithm. The resulting polynomial solution methods seem to be computationally interesting since they are able to better exploit specific structures and/or sparsity properties occurring in the symbol representation.

In the case $p > 2$ the Sylvester-like matrix is not generally a resultant matrix. The analysis of the relationships among the properties of the initial symbol and the properties of the associated Sylvester-like matrix is an interesting research field. In particular, it would be very interesting to relate the invertibility of $\mathcal{R}$ with some property about the distribution of the roots of the symbol in the complex plane. Furthermore, the possibility of devising effective polynomial solution methods for the case $p > 2$ is also an open issue. Finally, we recall that the Hurwitz type matrices $\mathcal{H}_k$ inherit a displacement structure. The interplay between this structure and the polynomial formulation resulting from the Bezout-like equation is under investigation.

5. REFERENCES

- [1] S. Barnett. Greatest common divisors from generalized Sylvester resultant matrices. *Linear and Multilinear Algebra*, 8(4):271–279, 1979/80.
- [2] C. Conti, L. Gemignani and L. Romani. From symmetric subdivision masks of Hurwitz type to interpolatory subdivision masks. *Linear Algebra Appl.*, 431(453):1971–1987, 2009.
- [3] G. Deslauriers and S. Dubuc. Symmetric iterative interpolation processes. *Constr. Approx.*, 5(1):49–68, 1989.
- [4] J. De Villiers and K. Hunter. On the construction and convergence of a class of symmetric interpolatory subdivision schemes. *East Journal on Approximations*, 12:151–188, 2006.
- [5] N. Dyn and D. Levin. Subdivision schemes in geometric modelling. *Acta Numer.*, 11:73–144, 2002.
- [6] G. Gentili and D. Struppa. Minimal degree solutions of polynomial equations. *Kybernetika*, 23:44–53, 1987.
- [7] M. F. Hassan and N. A. Dodgson. Ternary and three-point univariate subdivision schemes. In *Curve and*

- surface fitting (*Saint-Malo, 2002*), Mod. Methods Math., pages 199–208. Nashboro Press, Brentwood, TN, 2003.
- [8] R. Q. Jia. Interpolatory subdivision schemes induced by Box Splines. *Applied and Computational Harmonic Analysis*, 8:286–292, 2000.
 - [9] G. Li and W. Ma. A method for constructing interpolatory subdivision schemes and blending subdivision. *Computer Graphics Forum*, 26:185–201, 2007.
 - [10] J. Lian. On a -ary subdivision for curve design. I. 4-point and 6-point interpolatory schemes. *Appl. Appl. Math.*, 3:18–29, 2008.
 - [11] J. Lian. On a -ary subdivision for curve design. II. 3-point and 5-point interpolatory schemes. *Appl. Appl. Math.*, 3:176–187, 2008.
 - [12] C.A. Micchelli. Interpolatory subdivision schemes and wavelets. *J. Approx. Theory*, 86:41–71, 1996.
 - [13] G. Mustafa and F. Khan. A new 4-point C^3 quaternary approximating subdivision scheme. *Abstr. Appl. Anal.*, Art. ID 301967, 14 pages, 2009.
 - [14] M. Sabin. Subdivision of Box-splines. In Armin Iske, Ewald Quak and Michael S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, Mathematics and Visualization, pages 3–23. Springer, 2002.
 - [15] A. I. G. Vardulakis and P. N. R. Stoye. Generalized resultant theorem. *J. Inst. Math. Appl.*, 22(3):331–335, 1978.

Computing Loci of Rank Defects of Linear Matrices using Gröbner Bases and Applications to Cryptology

Jean-Charles Faugère, Mohab Safey El Din, Pierre-Jean Spaenlehauer
INRIA, Paris-Rocquencourt Center, SALSA Project
UPMC, Univ Paris 06, LIP6
CNRS, UMR 7606, LIP6
UFR Ingénierie 919, LIP6 Passy-Kennedy
Case 169, 4, Place Jussieu, F-75252 Paris
{Jean-Charles.Faugere,Mohab.Safey,Pierre-Jean.Spaenlehauer}@lip6.fr

ABSTRACT

Computing loci of rank defects of linear matrices (also called the MinRank problem) is a fundamental NP-hard problem of linear algebra which has applications in Cryptology, in Error Correcting Codes and in Geometry. Given a square linear matrix (i.e. a matrix whose entries are k -variate linear forms) of size n and an integer r , the problem is to find points such that the evaluation of the matrix has rank less than $r + 1$. The aim of the paper is to obtain the most efficient algorithm to solve this problem. To this end, we give the theoretical and practical complexity of computing Gröbner bases of two algebraic formulations of the MinRank problem. Both modelings lead to *structured algebraic systems*.

The first modeling, proposed by Kipnis and Shamir generates bi-homogeneous equations of bi-degree $(1, 1)$. The second one is classically obtained by the vanishing of the $(r + 1)$ -minors of the given matrix, giving rise to a determinantal ideal. In both cases, under genericity assumptions on the entries of the considered matrix, we give new bounds on the degree of regularity of the considered ideal which allows us to estimate the complexity of the whole Gröbner bases computations. For instance, the *exact* degree of regularity of the determinantal ideal formulation of a generic well-defined MinRank problem is $r(n - r) + 1$. We also give optimal degree bounds of the loci of rank defect which are reached under genericity assumptions; the new bounds are much lower than the standard multi-homogeneous Bézout bounds (or mixed volume of Newton polytopes).

As a by-product, we prove that the generic MinRank problem could be solved in polynomial time in n (when $n - r$ is fixed) as announced in a previous paper of Faugère, Levy-dit-Vehel and Perret. Moreover, using the determinantal ideal formulation, these results are used to break a cryptographic challenge (which was untractable so far) and allow us to evaluate precisely the security of the cryptosystem w.r.t. n , r and k . Our practical results suggest that, up to the software state of the art, this latter formulation is more adapted in the context of Gröbner bases computations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

Categories and Subject Descriptors

I.1.2 [Computing Methodologies]: Symbolic and Algebraic Manipulation—*Algorithms: Algebraic algorithms*; F.2.2 [Theory of Computation]: Analysis of algorithms and problem complexity—*Non numerical algorithms and problems: Geometrical problems and computation*; D.4.6 [Software]: Operating Systems—*Security and Protection: Cryptographic controls*

General Terms

Theory, Performance, Security

Keywords

Polynomial systems solving, Gröbner bases, Degree of regularity, Multi-homogeneous ideals, Determinantal ideals, Multivariate Cryptography, Generalized nonlinear Eigenvalue problem.

1. INTRODUCTION

Computing the locus of rank defect of a linear matrix (also called the MinRank problem) is of first importance for a wide range of applications. For instance, the security of many multivariate cryptosystems is closely related to the difficulty of solving MinRank problems [19, 7]. In geometry, the degeneracy locus of a projection of an algebraic surface defined by quadratic equations is the locus of rank defect of its jacobian matrix (which is a linear matrix) (see for instance [1]). Also, decoding metric rank codes can be reduced to a MinRank problem [21].

For $(n, k, r) \in \mathbb{N}^3$, we define the square MinRank problem as follows: given a square linear matrix of size n with k variables (i.e. a matrix whose entries are k -variate polynomials of degree 1 over a field $\mathbb{K}$), the goal is to find the locus of the points such that the matrix has a rank less than $r + 1$. This problem is difficult since deciding whether this locus is empty or not is NP-hard when $\mathbb{K}$ is a finite field [5]. When $k = 1$ the MinRank problem can be reduced to the EigenValue problem. Therefore, the MinRank problem can be seen as a generalized nonlinear EigenValue problem.

The ultimate objective of this paper is to find the most efficient method to solve this problem when the linear matrix is generic. In particular, we focus on two algebraic representations: the Kipnis-Shamir modeling [19] and the minors formulation.

Both representations are rather intuitive. For the Kipnis-Shamir modeling, the algebraic system is constructed by remarking that a matrix has a rank $\leq r$ if and only if there exist at least $n - r$ independent vectors in its kernel. Considering the coefficients of these vectors as variables gives rise to a quadratic system. On the other

hand, the minors modeling is obtained by considering all the minors of size $r+1$ of the linear matrix (which simultaneously vanish on the solutions of the MinRank problem).

Previous work. Since the MinRank problem has many applications, it has been extensively studied during the past decades, and a lot of different approaches have been tried (see [7] for details). So far, the most successful method seemed to be the Kipnis-Shamir formulation [19], which has been analyzed in [14]. Indeed, when combined with the algorithms F_5 [12] and FGLM [13], it can solve the challenges A and B proposed in [7]. However, the challenge C was remaining unbroken until now.

If $k = (n-r)^2$, then the number of solutions of a generic (n, k, r) -MinRank instance is finite and equal to the degree of the ideal I generated by the Kipnis-Shamir equations [14]. Since the solving strategy involves the FGLM algorithm (whose complexity is $O(\deg(I)^3)$), it is crucial to have good estimates of $\deg(I)$. The algebraic system obtained by the Kipnis-Shamir formulation is multi-homogeneous, thus upper bounds can be obtained by the multi-homogeneous Bézout number [14] or by computing the mixed volume of the associated Newton polytope [10]. However, the bounds provided by those techniques are not sharp.

Main results. The contributions of the paper are two-fold: theoretical and practical. Applying a Theorem from [15] to the Kipnis-Shamir modeling yields a bound on the degree of regularity of this system. From the viewpoint of the minors approach, we show that properties of the associated ideal are closely related to properties of determinantal ideals generated by minors of matrices whose entries are variables. More precisely, Lemma 1 brings out the relation between the ideal generated by the minors of a generic linear matrix and the ideal obtained by adding to a determinantal ideal n^2 generic linear forms. Thus properties known about determinantal ideals can be transferred to ideals corresponding to the minors modeling. In particular, this permits to establish explicit formulae for the exact degree of the ideal (Corollary 1) and for its Hilbert series (Theorem 3 and Theorem 4).

With this new information, the asymptotic complexity of solving the generic MinRank problem by both methods can be estimated, and it is shown (Section 4) that this complexity is polynomial in n when $k = (n-r)^2$ is constant. Surprisingly, using these new complexity estimates we found that the complexity bound of the minors approach is better than the complexity bound of the Kipnis-Shamir modeling.

Experiments were carried out with a view to checking the accuracy of the previous theoretical estimates. We apply those results to solve a cryptographic challenge based on MinRank which was untractable so far: experiments show that it is now possible to effectively break the challenge C from [7] by using the minors formulation and the F_5 algorithm in only 2^{49} arithmetic operations in $\text{GF}(65521)$.

Organization of the paper. After this short introduction, notations are introduced and the two modelings are formally defined. Some useful results are also recalled. Section 3 contains the main theoretical results and their proofs. Then, we derive complexity estimates of the cost of solving MinRank by using Gröbner bases algorithms. Finally, we present in Section 5 experimental results.

Acknowledgements. We wish to thank Ioannis Z. Emiris and Tomohiko Mizutani who provided bounds obtained by computing the mixed volume of the Newton polytope of the Kipnis-Shamir formulation. We are also grateful to Ludovic Perret for his helpful comments and suggestions. This work is supported by the EX-ACTA grant of the French National Research Agency (ANR-09-BLAN-0371-01) and the National Science Foundation of China.

2. PRELIMINARIES

General notations. Let $\mathbb{K}$ be a field. Let k, n and r be three integers, with $r < n$ and let $\mathbf{a} = (a_{1,1}^{(0)}, \dots, a_{n,n}^{(k)}) \in \mathbb{K}^{n^2(k+1)}$. Consider $\mathcal{M} \in \mathbb{M}_n(\mathbb{K}[x_1, \dots, x_k])$ the $n \times n$ linear matrix

$$\mathcal{M}_{i,j}(x_1, \dots, x_k) = a_{i,j}^{(0)} + \sum_{\ell=1}^k a_{i,j}^{(\ell)} x_\ell.$$

We called (n, k, r) -MinRank the problem of finding $(x_1, \dots, x_k)$ in $\overline{\mathbb{K}}^k$ (where $\overline{\mathbb{K}}$ denotes the algebraic closure of $\mathbb{K}$) such that the rank of $\mathcal{M}(x_1, \dots, x_k)$ is less than $r+1$.

In this paper, we focus on the generic case, i.e. when $\mathbf{a}$ is chosen “at random”. If $k = (n-r)^2$ (resp. $k < (n-r)^2$), the problem admits a finite number of solutions (see [14]) and is called *well-defined* (resp. *over-defined*). Note that if the problem is under-defined ($k > (n-r)^2$), it can be reduced to the well-defined case by specializing $k - (n-r)^2$ variables to random values [14].

An interesting subclass of problems is the homogeneous MinRank problem, obtained when $a_{i,j}^{(0)} = 0$ for all (i, j) .

The Kipnis-Shamir formulation. $(x_1, \dots, x_k)$ is solution of the (n, k, r) -MinRank problem if and only if there are at least $n-r$ independent vectors in the kernel of $\mathcal{M}(x_1, \dots, x_k)$. Since we assumed that $\mathbf{a}$ is chosen generically, we can suppose that a basis of the kernel can be written in systematic form [14]. Consider the following $n \times (n-r)$ matrix:

$$\mathcal{K} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ y_1^{(1)} & y_1^{(2)} & \dots & y_1^{(n-r)} \\ \vdots & \vdots & \ddots & \vdots \\ y_r^{(1)} & y_r^{(2)} & \dots & y_r^{(n-r)} \end{pmatrix}.$$

The Kipnis-Shamir modeling is constructed by considering the algebraic system $\mathcal{M} \cdot \mathcal{K} = 0$. Indeed, if $(x_1, \dots, x_k, y_1^{(1)}, \dots, y_r^{(n-r)})$ is a solution of the algebraic system, then $(x_1, \dots, x_k)$ is solution of the corresponding MinRank problem.

On the one hand, the system can be seen as a multi-homogeneous system with the following partition of variables:

$$\{x_1, \dots, x_k\} \cup \{y_1^{(1)}, \dots, y_r^{(1)}\} \cup \dots \cup \{y_1^{(n-r)}, \dots, y_r^{(n-r)}\}.$$

On the other hand, it can also be considered as a bilinear system with the partition of variables $X \cup Y$.

The minors formulation. $(x_1, \dots, x_k)$ is solution of a (n, k, r) -MinRank problem if and only if all minors of size $r+1$ of $\mathcal{M}$ simultaneously vanish on this point. Thus the minors modeling is obtained by considering the algebraic system of all minors of size $r+1$.

Solving strategy. For the well-defined problem, we use the following strategy (for both modelings): first compute a grevlex Gröbner basis of the ideal generated by the equations with the F_5 algorithm [12], then compute a Gröbner basis for the lex ordering by using FGLM [13]. For applications in Cryptology, $\mathbb{K}$ is a finite field and it is often known that a solution of the problem lies in $\mathbb{K}^k$. Then it is possible to combine this approach with an exhaustive search over s variables. For every possible values of s variables, we solve the resulting over-determined $(n, (n-r)^2 - s, r)$ -MinRank problem.

Previous works. The strategy for solving well-defined MinRank problems involves the FGLM algorithm. Its complexity is well

known: $O(\deg(I)^3)$ arithmetic operations, where $\deg(I)$ is the degree of the ideal generated by the equations (this degree is the same for both modelings). Therefore, sharp bounds on $\deg(I)$ are required to estimate the complexity of this step. So far, bounds on this degree are obtained by considering the multi-homogeneous structure of the Kipnis-Shamir formulation. A bound can be obtained with the multi-homogeneous Bézout number: $\deg(I) \leq \binom{n}{r}^{n-r}$ (see [14] for details). Newton polytope techniques [10] permit to achieve slightly sharper bounds, but require heavier computations. However, the gap between known bounds and the real degree is big. For instance, for the (6,9,3) problem, the degree of the ideal generated by either of the two modelings is 980, whereas the associated Bézout number is 8000 [14], and the mixed volume bound of the associated Newton polytope is 7340¹.

To estimate the complexity of the computation of the grevlex Gröbner basis, upper bounds on the so-called *degree of regularity* of the ideal generated by the equations are required. This value is the highest degree encountered during a Gröbner basis calculation with respect to a graded monomial ordering. The complexity of the whole Gröbner basis computation can be estimated by $O(M(d_{\text{reg}})^\omega)$ [3, 2], where $M(d_{\text{reg}})$ denotes the number of monomials of degree less than or equal to d_{reg} , and ω is the linear algebra constant ($2 \leq \omega \leq 3$). Recently, we showed in [15] a sharp bound on the degree of regularity of generic affine bilinear systems:

THEOREM 1. [15, Theorem 6.1] *For the grevlex ordering, the degree of regularity of a generic affine bilinear 0-dimensional system over $\mathbb{K}[X, Y]$ is upper bounded by $\min(\text{card}(X), \text{card}(Y)) + 1$.*

The Kipnis-Shamir algebraic modeling is a bilinear system, thus this bound can be applied: $d_{\text{reg}} \leq \min(k, (n-r)r) + 1$. In the case of well-defined instances, $k = (n-r)^2$ and thus $d_{\text{reg}} \leq \min((n-r)^2, (n-r)r) + 1$. In comparison, the classical Macaulay bound would yield an upper bound of $n(n-r) + 1$ [20].

In [15, Section 6.1], we also proposed a variant of the F_5 algorithm dedicated to multi-homogeneous systems. This variant could speed-up the computation of the Gröbner basis of the Kipnis-Shamir system. However, there is so far no efficient implementation of this algorithm.

Determinantal ideals. Properties of the minors modeling are strongly related to properties of *determinantal ideals* generated by minors of matrices whose entries are variables. In this paper, $\mathcal{D}$ denotes the ideal of $\mathbb{K}[v_{1,1}, \dots, v_{n,n}]$ generated by all minors of size $r+1$ of the following $n \times n$ matrix:

$$\begin{pmatrix} v_{1,1} & \dots & v_{1,n} \\ \vdots & \ddots & \vdots \\ v_{n,1} & \dots & v_{n,n} \end{pmatrix}.$$

Many results are known about the structure of the ideal $\mathcal{D}$.

THEOREM 2. [6, page 679] *The dimension of $\mathcal{D}$ is $(2n-r)r$, and its Hilbert series is*

$$\text{HS}(t) = \frac{\det A(t)}{t^{\binom{2}{2}}(1-t)^{(2n-r)r}},$$

where $A(t)$ is the $r \times r$ matrix defined by

$$A_{i,j}(t) = \sum_{\ell=0}^{n-\max(i,j)} \binom{n-i}{\ell} \binom{n-j}{\ell} t^\ell.$$

The following Proposition is a consequence of the Thom-Porteous formula. This question has been discussed by Giambelli, Harris-Tu

¹This value was provided to us by Ioannis Z. Emiris and Tomohiko Mizutani.

and Baker. A short proof of this formula can be found in [18, page 261].

PROPOSITION 1. [18, page 261] *The degree of the determinantal ideal $\mathcal{D}$ is*

$$\prod_{i=0}^{n-r-1} \frac{i!(n+i)!}{(n-1-i)!(n-r+i)!}$$

3. THEORETICAL ANALYSIS OF THE MINORS FORMULATION

Applications require efficient methods to solve the affine Min-Rank problem. However, we start by studying the homogeneous case. Indeed, the structure of the homogeneous problem is closely related to that of the affine case, and is easier to describe from a theoretical viewpoint.

Notations Throughout this paper, $\mathbf{a}$ denotes the set of kn^2 variables $\{a_{1,1}^{(1)}, \dots, a_{n,n}^{(k)}\}$, $\mathbf{b}$ is the set of kn^2 variables $\{b_{1,1}^{(1)}, \dots, b_{n,n}^{(k)}\}$ and $\mathbf{c}$ is the set of n^4 variables $\{c_{1,1}^{(1,1)}, \dots, c_{n,n}^{(n,n)}\}$. We consider the generic matrix $\mathcal{M} \in M_n(\mathbb{K}(\mathbf{a})[x_1, \dots, x_k])$ defined by

$$\mathcal{M}_{i,j} = \sum_{\ell=1}^k a_{i,j}^{(\ell)} x_\ell.$$

In the following, $\mathcal{I}$ denotes the ideal generated by all minors of size $r+1$ of $\mathcal{M}$. X (resp. V) denotes the set of variables $\{x_1, \dots, x_k\}$ (resp. $\{v_{1,1}, \dots, v_{n,n}\}$).

We would like to point out that the results of this section can be extended to the case where $\mathcal{M}$ is a non-square matrix.

3.1 The under-defined homogeneous case

In this part of the paper, we suppose that $k > (n-r)^2$. When $k \leq (n-r)^2$, the system is 0-dimensional and this case is discussed in Section 3.2.

DEFINITION 1.

- We denote by $\tilde{\mathcal{I}}$ the ideal of $\mathbb{K}(\mathbf{a})[X, V]$ defined by

$$\tilde{\mathcal{I}} = \mathcal{I} + \left\langle v_{i,j} - \sum_{\ell=1}^k a_{i,j}^{(\ell)} x_\ell \right\rangle_{1 \leq i,j \leq n}.$$

- For $\mathbf{a} = (a_{1,1}^{(1)}, \dots, a_{n,n}^{(k)}) \in \mathbb{K}^{n^2 k}$, the specialization morphism is denoted by $\varphi_{\mathbf{a}}$:

$$\begin{array}{ccc} \mathbb{K}(\mathbf{a}) & \rightarrow & \mathbb{K} \\ f(a_{1,1}^{(1)}, \dots, a_{n,n}^{(k)}) & \mapsto & f(a_{1,1}^{(1)}, \dots, a_{n,n}^{(k)}) \end{array}$$

- $\tilde{\mathcal{D}}$ denotes the ideal of $\mathbb{K}(\mathbf{b}, \mathbf{c})[X, V]$ defined by:

$$\tilde{\mathcal{D}} = \mathcal{D} + \langle g_{i,j} \rangle_{1 \leq i,j \leq n},$$

where $g_{i,j} = \sum_{\ell=1}^k b_{i,j}^{(\ell)} x_\ell + \sum_{1 \leq \ell_1, \ell_2 \leq n} c_{i,j}^{(\ell_1, \ell_2)} v_{\ell_1, \ell_2}$ are generic linear forms.

- For $(\mathbf{b}, \mathbf{c}) \in \mathbb{K}^{n^2 k} \times \mathbb{K}^{n^4}$, $\psi_{\mathbf{b}, \mathbf{c}}$ denotes the specialization morphism:

$$\begin{array}{ccc} \mathbb{K}(\mathbf{b}, \mathbf{c}) & \rightarrow & \mathbb{K} \\ f(\mathbf{b}, \mathbf{c}) & \mapsto & f(\mathbf{b}, \mathbf{c}) \end{array}$$

The following Lemma is one of the main tools of this Section: it shows how to transfer properties of $\mathcal{D}$ to the ideal $\tilde{\mathcal{I}}$ generated by the minors.

LEMMA 1. Let $\mathcal{P}$ be a property which holds on some ideals of $\mathbb{K}[X, V]$. Suppose that there exists a nonempty Zariski open set $O_P \subset \mathbb{K}^{n^2k} \times \mathbb{K}^{n^4}$ such that $\forall (\mathbf{b}, \mathbf{c}) \in O_P$, $\mathcal{P}$ is verified on $\psi_{\mathbf{b}, \mathbf{c}}(\tilde{\mathcal{D}})$. Then there exist nonempty Zariski open sets $O' \subset \mathbb{K}^{n^2k} \times \mathbb{K}^{n^4}$ and $O'' \subset \mathbb{K}^{n^2k}$ such that

$$\{\varphi_{\mathbf{a}}(\tilde{\mathcal{J}}) : \mathbf{a} \in O''\} = \{\psi_{\mathbf{b}, \mathbf{c}}(\tilde{\mathcal{D}}) : (\mathbf{b}, \mathbf{c}) \in O'\}$$

and the property $\mathcal{P}$ holds for every ideal in this set.

PROOF. Let F denote the complement of O_P in $\mathbb{K}^{n^2k} \times \mathbb{K}^{n^4}$, and let $I \subset \mathbb{K}[\mathbf{b}, \mathbf{c}]$ denote the ideal of polynomials vanishing on F . The property $\mathcal{P}$ holds on ideals and is independent on the set of generators. We want to encode this fact in our polynomial modeling. Consider the following $n^2 \times (n^2 + k)$ matrix:

$$\mathcal{C} = \begin{pmatrix} c_{1,1}^{(1,1)} & \dots & c_{1,1}^{(n,n)} & b_{1,1}^{(1)} & \dots & b_{1,1}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots \\ c_{1,n}^{(1,1)} & \dots & c_{1,n}^{(n,n)} & b_{1,n}^{(1)} & \dots & b_{1,n}^{(k)} \\ c_{2,1}^{(1,1)} & \dots & c_{2,1}^{(n,n)} & b_{2,1}^{(1)} & \dots & b_{2,1}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots \\ c_{n,n}^{(1,1)} & \dots & c_{n,n}^{(n,n)} & b_{n,n}^{(1)} & \dots & b_{n,n}^{(k)} \end{pmatrix}.$$

Each line of this matrix represents one generator $g_{i,j}$ of the ideal $\tilde{\mathcal{D}}$. If we replace this set of linear forms by an invertible linear combination of the $g_{i,j}$'s, then the ideal generated is the same, but the coefficients of the new generators do not necessarily belong to O_P . Thus we want to find a larger Zariski open set (named $\hat{O}$ in the sequel) such that if the coefficients of the $g_{i,j}$'s lie in $\hat{O}$, then the coefficients of any invertible linear combination of the $g_{i,j}$'s lie in $\hat{O}$.

For $M \in \text{GL}_{n^2}(\mathbb{K})$, let $I_M \subset \mathbb{K}[X, V]$ denote the ideal obtained by performing the linear change of variables $\mathcal{C}' = M \cdot \mathcal{C}$ and let F_M denote the variety of I_M . Since $\mathbb{K}[X, V]$ is Noetherian, the set $\bigcap_{M \in \text{GL}_{n^2}(\mathbb{K})} F_M$ is a Zariski closed subset. Let $\hat{O}$ be its complement. Then $\hat{O}$ is a nonempty Zariski open subset and $\forall (\mathbf{b}, \mathbf{c}) \in \hat{O}$, $\psi_{\mathbf{b}, \mathbf{c}}(\tilde{\mathcal{D}})$ verifies the property $\mathcal{P}$.

Let $h \in \mathbb{K}[\mathbf{b}, \mathbf{c}]$ be the determinant of the $n^2 \times n^2$ matrix of the n^2 first columns of $\mathcal{C}$. The inequation $h(\mathbf{b}, \mathbf{c}) \neq 0$ defines a nonempty Zariski open subset $O_{\det}$ of $\mathbb{K}^{n^2k} \times \mathbb{K}^{n^4}$. Let O' be equal to $\hat{O} \cap O_{\det}$. Then consider the vector $\mathbf{id} = (\text{id}_{i,j}^{(\ell_1, \ell_2)})$ defined by

$$\text{id}_{i,j}^{(\ell_1, \ell_2)} = \begin{cases} 1 & \text{if } (i, j) = (\ell_1, \ell_2) \\ 0 & \text{otherwise.} \end{cases}$$

Then let $O'' \subset \mathbb{K}^{n^2k}$ denote the set $\{\mathbf{a} : (\mathbf{a}, \mathbf{id}) \in O'\}$. Then O'' is a nonempty Zariski open subset of $\mathbb{K}^{n^2k}$.

Let $(\mathbf{b}, \mathbf{c})$ be in O' . Consequently, the $n^2 \times n^2$ matrix of the n^2 first columns of $\psi_{\mathbf{b}, \mathbf{c}}(\mathcal{C})$ is invertible, and thus, by performing a linear combination of the generators, there exists $\mathbf{a} \in \mathbb{K}^{n^2k}$ such that

$$\begin{aligned} & \left\langle \sum_{1 \leq \ell_1, \ell_2 \leq n} c_{i,j}^{(\ell_1, \ell_2)} v_{\ell_1, \ell_2} + \sum_{\ell=1}^k b_{i,j}^{(\ell)} x_{\ell} \right\rangle_{1 \leq i, j \leq n} \\ &= \left\langle v_{\ell_1, \ell_2} - \sum_{\ell=1}^k a_{i,j}^{(\ell)} x_{\ell} \right\rangle_{1 \leq i, j \leq n}. \end{aligned}$$

Then note that

$$\psi_{\mathbf{b}, \mathbf{c}}(\tilde{\mathcal{D}}) = \mathcal{D} + \left\langle v_{\ell_1, \ell_2} - \sum_{\ell=1}^k a_{i,j}^{(\ell)} x_{\ell} \right\rangle_{1 \leq i, j \leq n} = \varphi_{\mathbf{a}}(\tilde{\mathcal{J}}).$$

This shows the inclusion $\{\psi_{\mathbf{b}, \mathbf{c}}(\tilde{\mathcal{D}}) : (\mathbf{b}, \mathbf{c}) \in O'\} \subset \{\varphi_{\mathbf{a}}(\tilde{\mathcal{J}}) : \mathbf{a} \in O''\}$. Conversely, let $\mathbf{a}$ be in O'' . By construction, $(\mathbf{a}, \mathbf{id})$ is in O' and

$\varphi_{\mathbf{a}}(\tilde{\mathcal{J}}) = \psi_{\mathbf{a}, \mathbf{id}}(\tilde{\mathcal{D}})$. Thus $\{\varphi_{\mathbf{a}}(\tilde{\mathcal{J}}) : \mathbf{a} \in O''\} \subset \{\psi_{\mathbf{b}, \mathbf{c}}(\tilde{\mathcal{D}}) : (\mathbf{b}, \mathbf{c}) \in O'\}$. $\square$

In order to prove results on $\varphi_{\mathbf{a}}(\mathcal{J})$ (for generic $\mathbf{a}$), we use the following strategy:

- deduce properties of $\psi_{\mathbf{b}, \mathbf{c}}(\tilde{\mathcal{D}})$ by adding to $\mathcal{D}$ generic linear forms;
- with Lemma 1, transfer those properties to $\varphi_{\mathbf{a}}(\tilde{\mathcal{J}})$;
- finally, prove properties of $\varphi_{\mathbf{a}}(\mathcal{J})$ by eliminating the variables V .

From now on, we suppose that $n > 1$ and $\prec$ denotes the strict lexicographical ordering on $\mathbb{N}^2$: $(i_1, j_1) \prec (i_2, j_2)$ if and only if

$$\begin{cases} i_1 < i_2 \text{ or} \\ i_1 = i_2 \text{ and } j_1 < j_2. \end{cases}$$

We recall that $g_{i,j}$ is a generic linear form (see Definition 1).

PROPOSITION 2.

Denote by $\mathcal{D}_{\prec(i,j)}$ the ideal $\mathcal{D} + \langle g_{\ell_1, \ell_2} \rangle_{(\ell_1, \ell_2) \prec (i,j)} \subset \mathbb{K}[X, V]$. There exists a nonempty Zariski open subset $O \subset \mathbb{K}^{n^2k} \times \mathbb{K}^{n^4}$ such that, if $(\mathbf{b}, \mathbf{c}) \in O$, then for all $(i, j) \in \{1, \dots, n\}^2$, $\psi_{\mathbf{b}, \mathbf{c}}(g_{i,j})$ does not divide 0 in $\mathbb{K}[X, V] / \psi_{\mathbf{b}, \mathbf{c}}(\mathcal{D}_{\prec(i,j)})$.

PROOF. It is proved in [4, Theorem 2.10 and Remark 2.12], that $\mathcal{D}$ is a prime ideal. Moreover $\dim(\mathcal{D}) \geq 2$ (Theorem 2), thus there exists a nonempty Zariski open subset $O_{1,1}$ such that if $(\mathbf{b}, \mathbf{c}) \in O_{1,1}$, then $\psi_{\mathbf{b}, \mathbf{c}}(g_{1,1})$ does not divide 0 in $\mathbb{K}[X, V] / \mathcal{D}$. Furthermore, since $\mathcal{D}$ is prime, $\text{Spec}(\mathbb{K}[X, V] / \mathcal{D})$ is a reduced and irreducible scheme. According to [16, Corollary 3.4.14], cutting a reduced and irreducible scheme of dimension ≥ 2 by a generic hyperplane yields an irreducible and reduced scheme (it is a consequence of Bertini's First Theorem). Therefore, there exists a nonempty Zariski open subset $O'_{1,1}$ such that if $(\mathbf{b}, \mathbf{c}) \in O'_{1,1}$, then $\psi_{\mathbf{b}, \mathbf{c}}(\mathcal{D} + g_{1,1})$ is also radical and irreducible, thus prime. By induction, there exist nonempty Zariski open sets $O_{i,j}$ and $O'_{i,j}$ such that, if $(\mathbf{b}, \mathbf{c}) \in O_{i,j}$, then $\psi_{\mathbf{b}, \mathbf{c}}(g_{i,j})$ does not divide 0 in $\mathbb{K}[X, V] / \mathcal{D}_{\prec(i,j)}$, and if $(\mathbf{b}, \mathbf{c}) \in O'_{i,j}$, then $\psi_{\mathbf{b}, \mathbf{c}}(\mathcal{D}_{\prec(i,j)} + g_{i,j})$ is prime. Finally,

$$O = \bigcap_{(i,j) \in \{1, \dots, n\}^2} O_{i,j}$$

is the wanted nonempty Zariski open subset. $\square$

REMARK. We would like to point out that the condition $k > (n-r)^2$ is crucial for the proof of Proposition 2: this proof relies on Bertini's Theorem [16, Theorem 3.4.10], which is only valid if the projective dimension is ≥ 2 (i.e. the Krull dimension is ≥ 3). A consequence of this theorem is that if a prime homogeneous ideal has dimension $d \geq 3$, then adding $d-2$ generic linear forms yields a prime ideal of dimension 2 [16, Corollary 3.4.14]. Consequently, the maximum number of generic linear forms we can add such that each form does not divide zero in the previous quotient ring is $\dim(\mathcal{D}) + k - 1 = (2n-r)r + k - 1$. We need to add n^2 linear forms to define the generic MinRank problem and $(2n-r)r + k - 1 \geq n^2$ if and only if $k > (n-r)^2$.

COROLLARY 1. There exists a nonempty Zariski open subset O_1 of $\mathbb{K}^{n^2k}$ such that if $\mathbf{a} \in O_1$, then the dimension of $\varphi_{\mathbf{a}}(\mathcal{J})$ is $k - (n-r)^2$ and its degree is

$$\prod_{i=0}^{n-r-1} \frac{i!(n+i)!}{(n-1-i)!(n-r+i)!}.$$

PROOF. Consider $\mathcal{D}$ as an ideal of $\mathbb{K}[X, V]$. From Proposition 1, its degree is $\prod_{i=0}^{n-r-1} \frac{i!(n+i)!}{(n-1-i)!(n-r+i)!}$. From Theorem 2, the dimension of this ideal is $(2n-r)r + k$. According to Proposition 2, there

exists a nonempty Zariski open subset O of $\mathbb{K}^{n^2k} \times \mathbb{K}^{n^4}$ such that, $\psi_{\mathbf{b},\mathbf{c}}(\tilde{\mathcal{D}})$ has the same degree as $\mathcal{D}$ and its dimension is $k - (n-r)^2$ if $(\mathbf{b}, \mathbf{c}) \in O$ (since adding to an ideal a linear form which is not a divisor of zero in the quotient ring does not change the degree and decreases the dimension by 1).

Next, Lemma 1 shows that there exists a nonempty Zariski open subset $O_1 \subset \mathbb{K}^{n^2k}$, such that if $\mathbf{a} \in O_1$, then

$$\deg(\varphi_{\mathbf{a}}(\tilde{\mathcal{S}})) = \prod_{i=0}^{n-r-1} \frac{i!(n+i)!}{(n-1-i)!(n-r+i)!}.$$

Finally note that in $\varphi_{\mathbf{a}}(\tilde{\mathcal{S}})$, the variables V are linear combinations of the variables X . Thus

$$\begin{aligned} \deg(\varphi_{\mathbf{a}}(\tilde{\mathcal{S}})) &= \deg(\varphi_{\mathbf{a}}(\tilde{\mathcal{S}}) \cap \mathbb{K}[X]) \\ &= \deg(\varphi_{\mathbf{a}}(\mathcal{S})). \end{aligned} \quad \square$$

The Hilbert series is a useful tool to describe homogeneous ideals of $\mathbb{K}[X]$. If $I \subset \mathbb{K}[X]$, it is defined as follows:

$$\text{HS}(t) = \sum_{d \in \mathbb{N}} \dim(\mathbb{K}[X]_d / I_d) t^d,$$

where $\mathbb{K}[X]_d$ is the vector space of homogeneous polynomials of degree d and I_d denotes the vector space $I \cap \mathbb{K}[X]_d$.

Many information can be read off from this series. For instance, the dimension, the degree and the degree of regularity can be computed once this series is known. More precisely, if $\text{HS}(t) \in \mathbb{Z}[[t]]$ is the Hilbert series of an ideal $I \subset \mathbb{K}[X]$, then

- the smallest d such that $(1-t)^d \text{HS}(t)$ is a polynomial is the dimension of I ;
- if the dimension of I is 0, then the evaluation $\text{HS}(1)$ gives the degree of the ideal and $\deg(\text{HS}(t)) + 1$ is the degree of regularity of I .

The next theorem provides an explicit formula for the Hilbert series of the ideal generated by the minors of a generic linear matrix in the homogeneous under-defined case:

THEOREM 3. *There exists a nonempty Zariski open subset O_2 of $\mathbb{K}^{n^2k}$ such that if $\mathbf{a} \in O_2$, then the Hilbert series of $\varphi_{\mathbf{a}}(\mathcal{S})$ is*

$$\text{HS}_{\varphi_{\mathbf{a}}(\mathcal{S})}(t) = \frac{\det A(t)}{t^{\binom{r}{2}} (1-t)^{k-(n-r)^2}},$$

where $A(t)$ is the $r \times r$ matrix defined in Theorem 2.

PROOF. In [6, Corollary 1], it is shown that the Hilbert series of $\mathcal{D} \subset \mathbb{K}[V]$ is

$$\text{HS}_{\mathcal{D}}(t) = \frac{\det A(t)}{t^{\binom{r}{2}} (1-t)^{(2n-r)r}}.$$

Thus the Hilbert series of $\mathcal{D}$ as an ideal of $\mathbb{K}[X, V]$ is

$$\frac{\det A(t)}{t^{\binom{r}{2}} (1-t)^{(2n-r)r + \text{card}(X)}} = \frac{\det A(t)}{t^{\binom{r}{2}} (1-t)^{(2n-r)r+k}}.$$

Let O be the Zariski open set defined in Proposition 2. Adding to an ideal a linear form which is not a divisor of zero in the quotient ring multiplies the Hilbert series by $(1-t)$. Thus, if $(\mathbf{b}, \mathbf{c}) \in O$, then the Hilbert series of $\psi_{\mathbf{b},\mathbf{c}}(\tilde{\mathcal{D}})$ is

$$\frac{\det A(t)}{t^{\binom{r}{2}} (1-t)^{k-(n-r)^2}}.$$

Then, applying Lemma 1, the result can be transferred to $\varphi_{\mathbf{a}}(\tilde{\mathcal{S}})$ (for $\mathbf{a}$ in a nonempty Zariski open set O_2). Let G be a Gröbner basis of $\varphi_{\mathbf{a}}(\mathcal{S})$. Then

$$G \cup \{v_{i,j} - \sum_{\ell=1}^k a_{i,j}^{(\ell)} x_{\ell}\}_{1 \leq i,j \leq n}$$

is a Gröbner basis of $\varphi_{\mathbf{a}}(\tilde{\mathcal{S}})$ for a grevlex ordering with $V > X$ (i.e. a grevlex ordering such that $v_{i,j}^{(\ell_1, \ell_2)} > x_{\ell}$ for all $i, j, \ell, \ell_1, \ell_2$). Consequently, $\mathbb{K}[X]/\varphi_{\mathbf{a}}(\mathcal{S})$ is isomorphic (as $\mathbb{K}$ -vector spaces) to $\mathbb{K}[X, V]/\varphi_{\mathbf{a}}(\tilde{\mathcal{S}})$, thus the Hilbert series of $\varphi_{\mathbf{a}}(\tilde{\mathcal{S}})$ is the same as the Hilbert series of $\varphi_{\mathbf{a}}(\mathcal{S})$. $\square$

3.2 Well-defined and over-determined cases

In this part, $k \leq (n-r)^2$, and we still consider the homogeneous MinRank problem. First, we propose a variant of the Fröberg Conjecture [17], which describes the structure of the ideal obtained by adding to $\mathcal{D}$ more than $\dim(\mathcal{D}) - 1$ generic linear forms $g_{i,j}$ (as defined in Definition 1).

CONJECTURE 1. *We use the same notations as Proposition 2. Let $\mathcal{D}_{\prec(i,j),d}$ denote the vector space of homogeneous polynomials of degree d in $\mathcal{D}_{\prec(i,j)}$. Then there exists a nonempty Zariski open subset O_3 of $\mathbb{K}^{n^2k} \times \mathbb{K}^{n^4}$ such that, if $(\mathbf{b}, \mathbf{c}) \in O_3$, then $\forall (i, j) \in \{1, \dots, n\}^2, \forall d \in \mathbb{N}$, the linear map*

$$\begin{aligned} \psi_{\mathbf{b},\mathbf{c}}(\mathcal{D}_{\prec(i,j),d}) &\longrightarrow \psi_{\mathbf{b},\mathbf{c}}(\mathcal{D}_{\prec(i,j),d+1}) \\ f &\longmapsto f \cdot \psi_{\mathbf{b},\mathbf{c}}(g_{i,j}) \end{aligned}$$

is of maximal rank.

From now on, we use the following notation: for a series $S \in \mathbb{Z}[[t]]$, $[S]$ denotes the series obtained by truncating S at the first null or negative coefficient.

COROLLARY 2. *If Conjecture 1 is true, and if $(\mathbf{b}, \mathbf{c}) \in O_3$, then the Hilbert series of $\psi_{\mathbf{b},\mathbf{c}}(\mathcal{D}_{\prec(i,j)} + g_{i,j})$ is*

$$\left[(1-t) \text{HS}_{\psi_{\mathbf{b},\mathbf{c}}(\mathcal{D}_{\prec(i,j)})}(t) \right],$$

PROOF. In order to simplify the notations, I denotes the ideal $\psi_{\mathbf{b},\mathbf{c}}(\mathcal{D}_{\prec(i,j)})$ and I_d denotes the set of polynomials of I of degree d . Let $\times \psi_{\mathbf{b},\mathbf{c}}(g_{i,j})$ denote the multiplication by $\psi_{\mathbf{b},\mathbf{c}}(g_{i,j})$ and let $\text{ann}(\psi_{\mathbf{b},\mathbf{c}}(g_{i,j}))$ be the ideal $\{f \in \mathbb{K}[X, V] : f \psi_{\mathbf{b},\mathbf{c}}(g_{i,j}) \in I\}$. Consider the following exact sequence:

$$0 \rightarrow \text{ann}(\psi_{\mathbf{b},\mathbf{c}}(g_{i,j}))_d \rightarrow \mathbb{K}[X, V]_d / I_d \xrightarrow{\times \psi_{\mathbf{b},\mathbf{c}}(g_{i,j})} \mathbb{K}[X, V]_{d+1} / I_{d+1} \rightarrow \mathbb{K}[X, V]_{d+1} / (I + \psi_{\mathbf{b},\mathbf{c}}(g_{i,j}))_{d+1} \rightarrow 0.$$

According to Conjecture 1, the dimension of $\text{ann}(\psi_{\mathbf{b},\mathbf{c}}(g_{i,j}))_d$ is equal to $\max(0, \dim(\mathbb{K}[X, V]_d / I_d) - \dim(\mathbb{K}[X, V]_{d+1} / I_{d+1}))$. It is well known that the alternate sum of the dimensions of an exact sequence of vector spaces is 0. Therefore,

$$\begin{aligned} &\dim(\mathbb{K}[X, V]_{d+1} / (I + \psi_{\mathbf{b},\mathbf{c}}(g_{i,j}))_{d+1}) \\ &= \max(\dim(\mathbb{K}[X, V]_{d+1} / I_{d+1}) - \dim(\mathbb{K}[X, V]_d / I_d), 0). \end{aligned}$$

Multiplying this equation by t^{d+1} and summing over $d \in \mathbb{N}$ yields the claimed relation between the Hilbert series. $\square$

THEOREM 4. *If Conjecture 1 is true, then there exists a non-empty Zariski open subset O_4 of $\mathbb{K}^{n^2k}$ such that for each $\mathbf{a} \in O_4$, the Hilbert series of $\varphi_{\mathbf{a}}(I)$ is*

$$\text{HS}_{\varphi_{\mathbf{a}}(I)}(t) = \left[(1-t)^{(n-r)^2-k} \frac{\det A(t)}{t^{\binom{r}{2}}} \right],$$

where $A(t)$ is the $r \times r$ matrix defined in Theorem 2.

PROOF. Consider $\widehat{\mathcal{D}}$ the determinantal ideal on which we add only $(2n-r)r+k-1$ generic linear forms:

$$\widehat{\mathcal{D}} = \mathcal{D} + \left\langle \sum_{\ell=1}^k b_{i,j}^{(\ell)} x_k + \sum_{1 \leq \ell_1, \ell_2 \leq n} c_{i,j}^{(\ell_1, \ell_2)} v_{\ell_1, \ell_2} \right\rangle_{(i,j) \in S}$$

where $S \subset \{1, \dots, n\}^2$ and $\text{card}(S) = (2n-r)r+k-1$. Now take $(\mathbf{b}, \mathbf{c})$ in the nonempty Zariski open set $O \cap O_3$, (O is defined in Proposition 2, and O_3 is defined in Conjecture 1). Thus the Hilbert series of $\psi_{\mathbf{b}, \mathbf{c}}(\widehat{\mathcal{D}})$ is $\text{HS}_{\psi_{\mathbf{b}, \mathbf{c}}(\widehat{\mathcal{D}})}(t) = \frac{\det A(t)}{t^{\binom{2}{2}}(1-t)}$. Thus, adding the $n^2 - (2n-r)r - k + 1$ remaining linear forms, and applying Corollary 2 for each linear form, it is proved that the Hilbert series of $\psi_{\mathbf{b}, \mathbf{c}}(\widehat{\mathcal{D}})$ is $\left[(1-t) \left[(1-t) \left[\dots (1-t) \left[\frac{\det A(t)}{t^{\binom{2}{2}}(1-t)} \right] \right] \right] \right]$. It is easy to prove that if $S \in \mathbb{Z}[[t]]$ is a series such that $S(0) \geq 1$ (which is the case when S is an Hilbert series of an homogeneous ideal), then $[(1-t)[S]] = [(1-t)S]$. Thus the Hilbert series of $\psi_{\mathbf{b}, \mathbf{c}}(\widehat{\mathcal{D}})$ can be rewritten as $\left[(1-t)^{(n-r)^2-k} \frac{\det A(t)}{t^{\binom{2}{2}}} \right]$.

Finally, by the same argument as in the proof of Theorem 3 (i.e. by using Lemma 1 and then eliminating the variables V), there exists a nonempty Zariski open set $O_4 \subset \mathbb{K}^{n^2k}$ such that, if $a \in O_4$, then the Hilbert series of $\varphi_{\mathbf{a}}(\mathcal{S})$ is the same. $\square$

The *degree of regularity* is a sharp indicator of the complexity of Gröbner basis algorithms. It is the highest degree of the polynomials occurring during the Gröbner basis computation. If I is a 0-dimensional homogeneous ideal, d_{reg} is precisely the lowest integer such that all monomials of degree d_{reg} are in I and can be read off from the Hilbert series (which is a polynomial):

$$d_{\text{reg}} = 1 + \deg(\text{HS}(t)).$$

Most bounds of the complexity of Gröbner basis algorithms (for instance F_4 [11] or F_5 [12]) are exponential in the degree of regularity. Therefore it is crucial to obtain sharp estimates of d_{reg} .

COROLLARY 3. *Under the same conditions as Theorem 4, the degree of regularity of $\varphi_{\mathbf{a}}(\mathcal{S})$ is*

$$\deg \left(\left[(1-t)^{(n-r)^2-k} \frac{\det A(t)}{t^{\binom{2}{2}}} \right] \right) + 1.$$

PROOF. The degree of regularity of a 0-dimensional homogeneous system is equal to the degree of the Hilbert series (given by Theorem 4) of the associated ideal plus 1. $\square$

COROLLARY 4. *The degree of regularity of the ideal generated by the minors formulation of a generic well-defined MinRank problem (i.e. $k = (n-r)^2$) is bounded by $d_{\text{reg}} \leq r(n-r) + 1$.*

PROOF. According to Corollary 3,

$$d_{\text{reg}} = \deg \left(\left[\frac{\det A(t)}{t^{\binom{2}{2}}} \right] \right) + 1 = \deg \left(\frac{\det A(t)}{t^{\binom{2}{2}}} \right) + 1.$$

On each row of the matrix $A(t)$, a polynomial with the highest degree is on the diagonal. Moreover, $\deg(A_{i,i}(t)) = n - i$. Thus

$$\deg(\det A(t)) \leq \sum_{i=1}^r (n-i) = nr - \frac{r(r+1)}{2}.$$

Finally $\text{HS}(t) = \frac{\det A(t)}{t^{\binom{2}{2}}}$, and

$$\begin{aligned} d_{\text{reg}} &= \deg(\text{HS}(t)) + 1 \\ &\leq nr - \frac{r(r+1)}{2} - \frac{r(r-1)}{2} + 1 \\ &= r(n-r) + 1. \quad \square \end{aligned}$$

This bound is sharp in practice: if the MinRank instance is generic, then the degree of regularity of the ideal generated by the minors is exactly $r(n-r) + 1$.

The affine well-defined and over-determined MinRank problem. In most applications, the MinRank problems occurring are affine. The analysis performed for the homogeneous MinRank problem permits to estimate the complexity of solving MinRank by the minors approach in the 0-dimensional affine case. Indeed, the maximal degree reached during the Gröbner basis computation is upper bounded by the degree of regularity of the ideal generated by the homogeneous parts of highest degree of the minors.

Therefore, the degree of regularity of the minors formulation of a generic affine (n, r, k) -MinRank problem is less or equal than the degree of regularity of the minors formulation of a generic homogeneous (n, r, k) -MinRank (given by Corollary 3). In practice, this bound is sharp: when the MinRank instance is generic, it is an equality.

4. COMPLEXITY ANALYSIS

In this section, we estimate the costs of the Gröbner basis computations and of the FGLM algorithm for generic well-defined $(k = (n-r)^2)$ affine MinRank problems.

4.1 The Kipnis-Shamir formulation

The arithmetic complexity of the F_5 algorithm [12] for computing a grevlex Gröbner basis can be estimated by $O(M(d_{\text{reg}})^\omega)$ [2, 3], where $M(d_{\text{reg}})$ denotes the number of monomials of degree less than or equal to d_{reg} and ω is the linear algebra constant.

We make the assumption that the Kipnis-Shamir modeling applied to a MinRank problem where the matrices are chosen generically leads to a generic enough bilinear system such that Theorem 1 holds. This assumption is verified experimentally.

Therefore, we get $d_{\text{reg}} \leq \min(k, (n-r)r) + 1$ and the complexity is upper bounded by $O\left(\binom{k+r(n-r)+d_{\text{reg}}}{d_{\text{reg}}}^\omega\right)$.

In applications, $r' = (n-r)$ is often constant, $k = (n-r)^2$, and we want to estimate the asymptotic complexity when n grows. According to Theorem 1, $d_{\text{reg}} = r'^2 + 1$ when n is big enough.

A straightforward computation gives

$$\binom{k+(n-r')r'+r'^2+1}{r'^2+1}^\omega \underset{n \rightarrow \infty}{\sim} \left(\frac{1}{(r'^2+1)!} \right)^\omega (k+nr')^\omega (r'^2+1) = O(n^{\omega(r'^2+1)}).$$

This estimate of the complexity is for standard Gröbner basis algorithms F_4 and F_5 for homogeneous systems. In [15, Section 6.1], a variant of F_5 dedicated to multi-homogeneous is proposed. The key observation is that the multi-homogeneous structure of the system induces a structure in the matrices occurring in the F_4 and F_5 algorithms. Consequently, those matrices can be decomposed into smaller matrices, whose row echelon forms can be computed independently. A consequence of this decomposition would be a speed-up and a reduction of the required memory. Since the Kipnis-Shamir modeling has a multi-homogeneous structure, this variant of F_5 could lead to practical improvements. However, so far there is no efficient implementation of this multi-homogeneous variant, and no precise complexity analysis.

4.2 The minors formulation

In this part, we estimate the asymptotic complexity of computing a grevlex Gröbner basis in the well-defined case ($k = (n-r)^2$). In particular, we fix $r' = (n-r)$, and we estimate the arithmetic complexity when n grows. As in Section 4.1, the complexity of the F_5 algorithm can be estimated by $O(M(d_{\text{reg}})^\omega)$.

According to Corollary 4, the complexity is then upper bounded by $O\left(\binom{k+r'(n-r')+1}{r'(n-r')+1}\right)^\omega$. An equivalent when n grows is

$$\left(\frac{k+r'(n-r')+1}{r'(n-r')+1}\right)^\omega \underset{n \rightarrow \infty}{\sim} \left(\frac{1}{k!}\right)^\omega (k+r'n)^{\omega k} = O(n^{\omega r^2}).$$

One observes that – in the well-defined case – the complexity bound of the minors approach is slightly better than the complexity bound of the Kipnis-Shamir modeling.

4.3 Complexity of FGLM in the well-defined case

With both modelings, when a grevlex basis is computed in the well-defined case ($k = (n-r)^2$), a change of ordering is required to obtain the lexicographical basis which gives the solutions of the problem. Corollary 1 yields the degree of the ideal (with the Kipnis-Shamir modeling or with the minors modeling). The complexity of FGLM is $O(\deg(I)^3)$, thus we need the asymptotic behaviour of the degree to perform a complexity analysis.

When $r' = n - r$ is constant, applying Corollary 1, we get

$$\deg(I) = \prod_{i=0}^{r'-1} \frac{(n+i)!}{(n-1-i)!} \cdot \prod_{i=0}^{r'-1} \frac{i!}{(r'+i)!} \underset{n \rightarrow \infty}{\sim} n^{r^2} \prod_{i=0}^{r'-1} \frac{i!}{(r'+i)!}.$$

Therefore, the asymptotic complexity of FGLM is $O(n^{3r^2})$.

5. EXPERIMENTAL RESULTS AND APPLICATIONS

In this Section, $\mathbb{K}$ is the finite field $\text{GF}(65521)$.

Workstation. Experimental results have been obtained with 24 Xeon quadricore processors 3.2 GHz, with 64 GB of RAM.

5.1 Computing the minors

The minors modeling raises questions about how to generate the equations. It is not clear how to compute efficiently all minors of size $r+1$ of a big matrix. For a $n \times n$ matrix, there are $\binom{n}{r+1}^2$ such minors, and each is a polynomial of degree $r+1$ in k variables. For instance, for an affine problem with $\mathbb{K} = \text{GF}(65521)$, $n = 11$, $k = 9$ and $r = 8$, it took 14 days on one CPU (with Maple). Fortunately, this computation can be parallelized: with 120 processes running simultaneously on 24 CPU, the computation lasted 12 hours. The size of the resulting algebraic system is 3466 MB.

For this computation, we used naive algorithms (each determinant was computed independently) but we believe that there is room for improvement by using more sophisticated algorithms.

5.2 The well-defined case

Here, $k = (n-r)^2$ and the ground field is $\mathbb{K} = \text{GF}(65521)$. This set of parameters is used in a MinRank-based authentication scheme [7].

Generation of the instances. For $(n, k, r) \in \mathbb{N}^3$, we generate a $n \times n$ matrix $M = (M_{i,j})$ where the $M_{i,j}$ are affine linear forms in k variables: $M_{i,j} = a_{i,j}^{(0)} + \sum_{\ell=1}^k a_{i,j}^{(\ell)} x_\ell$, where the $a_{i,j}^{(\ell)}$ are chosen uniformly at random in $\text{GF}(65521)$.

Interpretation of the results. Table 1 describes experimental results, for different values of the triplet (n, r, k) . In particular, we consider sets of parameters used in Cryptology for a MinRank-based authentication scheme [7]. The complexity of solving the MinRank problem is then directly related to the security of this cryptosystem. The values in italic font were not computed, but are estimates of the complexity based on the theoretical results from the previous section.

Chall.	A	B				C
	(6, 9, 3)	(7, 9, 4)	(8, 9, 5)	(9, 9, 6)	(10, 9, 7)	(11, 9, 8)
degree	980	4116	14112	41580	108900	259545
MH Bézout	8000	42875	175616	592704	1728000	4492125
Minors						
F_5 time	1.1s	37s	935s	18122s	229094s	2570396s
F_5 mem	488 MB	587 MB	1213 MB	5048 MB	25719MB	
F_4 Magma	4.6s	142.8s	3343.5s	∞		
d_{reg}	10	13	16	19	22	25
Nb op.	21.5	25.9	29.2	32.7	35.2	40.2
FGLM time	1.7s	97.2s	∞			
Kipnis-Shamir						
F_5 time	30s	3795s	328233s	∞		
F_5 mem	407 MB	3113 MB	58587 MB			
F_4 Magma	300s	48745s	∞			
d_{reg}	5	6	7			
Nb op.	30.5	37.1	43.4	50.4	57.4	64.4
FGLM time	35s	2580s	∞			

Table 1: Authentication scheme parameters

The row “degree” provides the degree of the ideal (i.e. the number of solutions in the algebraic closure) and can be compared with the multi-homogeneous Bézout bound (“MH Bézout”). The row “ F_5 time” (resp. “ F_5 mem”) gives the time (resp. the memory) needed to compute the grevlex Gröbner basis of the ideal under consideration. The computation is done with the F_5 algorithm from the FGB package. We also give the time obtained for the same Gröbner basis computations with the implementation of F_4 in Magma2.16, so that experiments can be reproduced. “ d_{reg} ” gives the degree of regularity of the ideal. Finally “Nb op.” indicates the logarithm (in base 2) of the exact number of arithmetic operations performed during the execution of the F_5 algorithm, and “FGLM time” provides the running time of FGLM (from the FGB package).

Note that the degree of regularity of the ideal generated by the minors matches the value given in Corollary 4. Moreover, note that the degree of the ideal is equal to the value provided by Corollary 1.

Looking at the logarithm of the number of arithmetic operations which is growing linearly, it seems clear that, for both formulations, the Gröbner basis computation is polynomial in n when $n - r$ is fixed, as announced in [14] and proved in this paper (Section 4).

We would like to emphasize that the FGLM step costs sometimes more than the grevlex Gröbner basis computation. In order to avoid this cost, a possible strategy is to combine the minors approach with an exhaustive search over some variables.

5.3 Solving the challenge C of the Courtois authentication scheme

Solving the challenge C requires to find one solution of a generic affine $(11, 9, 8)$ -MinRank problem which has a particularity: it is known that there is a solution $(x_1, \dots, x_9) \in \text{GF}(65521)^9$ in the ground field. Therefore we can combine the minors formulation with a partial exhaustive search. To this end, we specialize s variables and solve the corresponding over-determined $(11, 9 - s, 8)$ -MinRank problem for all specializations of the s variables. The degree of regularity of the over-determined systems can be estimated with Corollary 3, so the complexity of the complete computation can be approximated. For these systems, the degree of the ideal is 0 or 1. Consequently, a grevlex Gröbner basis is also a lex Gröbner basis and the FGLM algorithm is no longer required.

Table 2 shows the experimental results for different values of s . The row “ d_{reg} ” gives the degree of regularity obtained for each specialization of the s variables. The row “Nb op.” gives an estimate of the logarithm in base 2 of total number of operations needed to

		$(n = 11, k = 9 - s, r = 8)$			
	s	3	2	1	0
Minors	F_5 time	79s	1594s	80255s	<i>2570396s</i>
	F_5 mem	<1000 MB	2400 MB	29929 MB	
	d_{reg}	9	10	13	<i>25</i>
	Nb op.	73	60	49.1	<i>40.2</i>
KS	F_5 FGb	57000s	∞		
	F_5 mem	10539 MB			
	d_{reg}	7			
	Nb op.	88.6			

Table 2: Challenge C of the Courtois authentication scheme.

solve the challenge C. It is equal to $\log_2(65521^s \text{OpF}_5)$ where OpF_5 is the number of arithmetic operations used by the F_5 algorithm to solve one $(11, 9 - s, 8)$ -MinRank problem. The values in italic font were not effectively computed but are given as estimates based on practical and theoretical results.

First of all, we want to emphasize the fact that the degree of regularity of the ideal generated by the minors matches the one deduced from the generic Hilbert series (Corollary 3) in the over-determined case.

According to Table 2, the best practical choice seems to be $s = 1$. In practice, the 65521 computations of the over-determined systems can be parallelized, and the total number of required arithmetic operations ($2^{49.1}$) is quite practical. We estimate to 238 days the time needed to effectively solve this challenge on 64 quadricore processors. Therefore, the authentication scheme cannot be considered secure anymore with the set of parameters $(n = 11, k = 9, r = 8)$.

Note that it may be possible to compute directly a Gröbner basis of the ideal generated by the minors ($s = 0$). By interpolating the practical results, we give a rough estimate of the complexity of this computation: it would take approximately 29 days (on one CPU). However, it is not clear how much memory would be required, and the FGLM step could be untractable since the degree of the ideal is 259545 (Corollary 1).

6. CONCLUSION

In this paper, we studied two formulations of the MinRank problem from the viewpoint of efficiency and practical applications. In particular, the analysis of the ideals generated by the minors gave new information about the intrinsic structure of this problem.

Results from algebraic geometry about determinantal ideals permit to obtain the number of solutions for a generic MinRank problem when $k = (n - r)^2$. This value is important for the study of the complexity of the solving process since it has a direct impact on the complexity of FGLM.

We provided the Hilbert series and an explicit formula for the degree of regularity of the ideal generated by the minors. This information leads to a complexity analysis of the whole Gröbner basis computation. We also proposed a method to break the challenge C of the MinRank authentication scheme faster than any other known approaches. This method is feasible in practice since it requires only 2^{49} arithmetic operations.

Many interesting questions have arisen from this study. First, to be able to apply the minors approach on huge over-determined MinRank instances, algorithms for computing efficiently all the minors of size $r + 1$ of a linear matrix are required. Another question is to find how the multi-homogeneous structure of the Kipnis-Shamir formulation can be used to speed-up the computations, and to evaluate precisely its cost. We derived a formula from [15] to bound the

degree of regularity of the Kipnis-Shamir modeling. Although this bound is much sharper than any other known bounds, there is still a small gap between it and the real degree of regularity.

7. REFERENCES

- [1] B. Bank, M. Giusti, J. Heintz, M. Safey El Din, and E. Schost. On the geometry of polar varieties. *Applicable Algebra in Engineering, Communication and Computing*, 21(1):33–83, 2010.
- [2] M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004.
- [3] M. Bardet, J.-C. Faugère, B. Salvy, and B. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *Proceedings of MEGA*, 2005.
- [4] W. Bruns and U. Vetter. *Determinantal rings*. Springer, 1988.
- [5] J. Buss, G. Frandsen, and J. Shallit. The computational complexity of some problems of linear algebra. *Journal of Computer and System Sciences*, 58(3):572–596, 1999.
- [6] A. Conca and J. Herzog. On the Hilbert function of determinantal rings and their canonical module. *Proceedings of the American Mathematical Society*, pages 677–681, 1994.
- [7] N. Courtois. Efficient zero-knowledge authentication based on a linear algebra problem MinRank. In *Advances in Cryptology - Asiacrypt 2001*, volume 2248 of *LNCS*, pages 402–421. Springer.
- [8] D. Cox, J. Little, and D. O’Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer, 1997.
- [9] D. Eisenbud. *Commutative algebra with a view toward algebraic geometry*. Springer, 2004.
- [10] I. Emiris and J. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *Journal of Symbolic Computation*, 20(2):117–149, 1995.
- [11] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
- [12] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 75–83. ACM, 2002.
- [13] J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
- [14] J.-C. Faugère, F. Levy-dit Vehel, and L. Perret. Cryptanalysis of MinRank. In *Proceedings of the 28th Annual conference on Cryptology: Advances in Cryptology*, page 296. Springer, 2008.
- [15] J.-C. Faugère, M. Safey El Din, and P.-J. Spaenlehauer. Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree $(1,1)$: Algorithms and complexity. *arXiv:1001.4004v1 [cs.SC]*, 2010.
- [16] H. Flenner, L. van Gastel, and W. Vogel. *Joins and intersections*. Springer, 1991.
- [17] R. Fröberg. An inequality for Hilbert series of graded algebras. *Math. Scand.*, 56(2):117–144, 1985.
- [18] W. Fulton. *Intersection theory*. Springer, 1984.
- [19] A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In *Advances in Cryptology - CRYPTO’99*, volume 1666 of *LNCS*, pages 19–30. Springer, 1999.
- [20] D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *Computer Algebra, EUROCAL’83*, volume 162 of *LNCS*, pages 146–156. Springer, 1983.
- [21] A.-V. Ourivski and T. Johansson. New technique for decoding codes in the rank metric and its cryptography applications. *Problems of Information Transmission*, 38(3):237–246, 2002.

Output-Sensitive Decoding for Redundant Residue Systems

Majid Khonji
majid.khonji@imag.fr

Clément Pernet
clement.pernet@imag.fr

Jean-Louis Roch
jean-louis.roch@imag.fr

Thomas Roche
thomas.roche@imag.fr

Thomas Stalinski
thomas.stalinski@imag.fr

Grenoble Univ. INRIA MOAIS, LIG, 51, avenue J. Kuntzmann,
F38330 Montbonnot Saint Martin, France

ABSTRACT

We study *algorithm based fault tolerance* techniques for supporting malicious errors in distributed computations based on Chinese remainder theorem. The description holds for both computations with integers or with polynomials over a field. It unifies the approaches of redundant residue number systems and redundant polynomial systems through the Reed Solomon decoding algorithm proposed by Gao. We propose several variations on the application of the extended Euclid algorithm, where the error correction rate is adaptive. Several improvements are studied, including the use of various criteria for the termination of the Euclidean Algorithm, and an acceleration using the Half-GCD techniques. When there is some redundancy in the input, a gap in the quotient sequence is stated at the step matching the error correction, which enables early termination parallel computations. Experiments are shown to compare these approaches.

Categories and Subject Descriptors

G.4 [Mathematics and Computing]: Mathematical Software—*Algorithm Design and Analysis, Reliability and robustness*; E.4 [Coding and Information Theory]: [Error Control Codes]; I.1.2 [Computing Methodologies]: Symbolic and Algebraic Manipulation

General Terms

Algorithms, Security

Keywords

Algorithm Based Fault Tolerance, Redundant Residue Number System, Early termination, Adaptive algorithms, Fast extended Euclidean algorithm

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

1. INTRODUCTION

In the context of distributed computations, grid computing or more recently cloud computing, the computation has to be secured against failures of several kinds: crash fault, where an expected data is never received, or malicious errors where a data is still being transmitted but is corrupted (e.g. by a malicious code). The model of Byzantine errors [14, 18] illustrates the case where the computation is not always corrupted, and therefore, one might still want to use the results provided by this computing node. These errors can be managed, at least heuristically, using error detection: for instance blacklisting of corrupted machines, or checking post conditions on the results. Yet, errors can be corrected using fault tolerant techniques based on redundancy: either by duplicating the same computation on several machines (e.g. using replication codes); or, to further improve efficiency, by introducing redundancy at the algorithm level in the framework of Algorithm Based Fault Tolerance (ABFT) [10]. For instance, to manage some crash faults, ABFT matrix-vector computation is achieved in [2]: redundancy is added in the matrix-vector product by slightly increasing the dimension of the matrix based on an error correcting code.

Evaluation/interpolation techniques, such as Chinese remaindering algorithm, are very well suited for parallel computations, as they allow to split a computation into independent tasks. For example, the computation of the determinant of an integer matrix, can be parallelized with a residue number system (RNS), where each parallel task is the computation of a determinant modulo a prime number.

When the result to be reconstructed by a RNS is expected to be small, but no bound is known, the early termination approach [4] allows to limit the number of modular computations to the appropriate amount. This adaptive approach lead to a complexity that is output-sensitive. It is of great interest for e.g. computations with sparse or structured matrices.

Residue systems also allow to easily introduce redundancy, by doing additional modular computations, in order to form a redundant residue number system (RRNS) [21, 15]. But they heavily rely on an a priori knowledge of a bound on the output, thus preventing the use of early termination.

In this paper, we propose a unified point of view on redundant residue systems for both polynomial and integer computations, gathering together Gao's algorithm [5] for the polynomials and Mandelbaum's algorithm [16] for the integers. We then relax the prerequisites for these algorithms

to make them output sensitive. The advantage is twofold: the computation is adaptive in both the output size and the error rate induced by the varying reliability of a global computing environment. We also describe how to incorporate a fast Euclidean algorithm [20] in this framework.

2. REDUNDANT RESIDUE SYSTEMS

2.1 Notations and state of the art

Let R be an Euclidean ring equipped with an Euclidean function $\nu : R \rightarrow \mathbb{Z}_+$. We will assume that ν is multiplicative and sub-additive, i.e. for all $a, b \in R$:

$$\begin{aligned} \nu(ab) &= \nu(a)\nu(b), \\ \nu(a+b) &\leq \nu(a) + \nu(b). \end{aligned}$$

For $R = \mathbb{Z}$, the usual Euclidean function $\nu(x) = |x|$ can be used. For $R = K[X]$, where K is a field, one can use the function $\nu(a) = 2^{\deg(a)}$ (see [6, Note 3.1]). For an element $x \in R$, we define its *amplitude* as the valuation $\nu(x)$ and its *size* as the logarithm of its amplitude $\log_2 \nu(x)$. Note that the size is a real number.

Let $P_1, \dots, P_m$ be m elements in R pairwise relatively prime. Define $\Pi = \prod_{i=1}^m P_i$.

The usual construction of redundant residue system (or ideal error correcting codes) proceeds as follows: consider $\mathcal{M} = \{X \in R \mid \nu(X) < \kappa\}$ with $0 \leq \kappa \leq \nu(\Pi)$. The code $\mathcal{C}$ is a subset of $R/(P_1) \times \dots \times R/(P_m)$ defined by $\mathcal{C} = \{(p \bmod P_1, \dots, p \bmod P_m) \mid p \in \mathcal{M}\}$. This allows to easily define the notion of symbol of the code (each residue of p modulo a P_i). In the literature [16, 7, 13] the P_i are usually sorted by increasing valuation $\nu(P_i)$ and κ is defined as $\kappa = \nu\left(\prod_{i=1}^k P_i\right)$ for $0 < k \leq m$. The redundant symbols are the residues modulo the P_i 's for $k < i \leq m$. Defining the distance between $x = (x[P_1], \dots, x[P_m])$ and $y = (y[P_1], \dots, y[P_m])$, as the number of non-zero coordinates of $x - y$ (i.e. the Hamming weight of the support of $x - y$), one can prove that this code has minimal distance $m - k + 1$ [16, Theorem 1]. The code is therefore $\lfloor \frac{m-k}{2} \rfloor$ -corrector in theory.

Now finding a polynomial time algorithm decoding up to this bound has been a complicated story for the case $R = \mathbb{Z}$. Mandelbaum [16] uses a perturbation technique, and requires that $P_1, \dots, P_n$ remain relatively close (with $p_i = \nu(P_i)$). In [7], a unique decoding algorithm is given for up to

$$t \leq \frac{(m-k) \log P_1}{\log P_1 + \log P_m}$$

errors, where $P_1 < \dots < P_m$. This highlights a specificity of redundant residue number systems, that some residues provide more information if their modulo is large. Thus the errors are weighted depending on the symbol on which they occur, and one is tempted to think that it will make the $\lfloor \frac{m-k}{2} \rfloor$ error correction rate unreachable. Now interestingly, Guruswami & Al. [9] have shown that this is not the case, since the heterogeneous distribution of weight has already been accounted for, when computing the distance (as the primes are sorted by increasing order). They can consequently produce the first algorithm that matches the $\frac{m-k}{2}$ error correction capacity.

These algorithms all require that the residues are all available when the correction starts, and that they are sorted by

increasing valuation of the corresponding modulo. In the context of distributed or cloud computations, we wish to be able to treat them *as they come* in order to avoid synchronizations. We also want to permit early termination: the Chinese remaindering reconstruction and decoding are stopped whenever a *valid* result is detected.

In this section, we will present an adaptation of the usual decoding algorithm (originating from [16, 5]), where no order on the moduli is assumed and where the parameters k and t of the code are unknown. In order to relax these assumptions we move away from the standard description of the code, where information is described by a vector of residues, and prefer to consider the reconstructed values in the main ring R . The Chinese remainder theorem states that these descriptions are equivalent. Our contribution only consists in the introduction of a different metric on the Euclidean ring R , that fits more naturally the decoding algorithm (algorithm 1) based on extended Euclidean algorithm: it allows to describe the decoding algorithm in a unified manner over the Euclidean rings of polynomials and integers, and without conditions on the size and order of the moduli. After the description of a first *bounded error* decoding algorithm (algorithm 1), we propose an improvement making it adaptive in the error correction capacity in algorithms 2 and 3.

The following notations will be used throughout the sequel. X is a codeword and Y is the erroneous word received. Then $E = Y - X$ is the error. We define $I = \{i \in 1 \dots m \mid X \neq Y \bmod P_i\}$ as the set of indices of the erroneous residues. The *impact* Π_F of the error is defined by $\Pi_F = \prod_{i \in I} P_i$ and we denote $\Pi_T = \Pi / \Pi_F = \prod_{i \notin I} P_i$.

We will define an *amplitude-code* as a subset $\mathcal{C} = \{x \in R \mid \nu(x) < \kappa\}$ of the set $\mathcal{A} = \{x \in R \mid \nu(x) < \nu(\Pi)\}$. In order to form an error correcting code, there remain to define a distance in $\mathcal{A}$.

LEMMA 2.1. *The function*

$$\begin{aligned} \Delta : \mathcal{A} \times \mathcal{A} &\rightarrow \mathbb{R}_+ \\ (x, y) &\mapsto \sum_{i \mid x \neq y \bmod P_i} \log_2 \nu(P_i) \end{aligned}$$

is a metric over $\mathcal{A}$.

PROOF. Δ is positive, symmetric, satisfies the triangular inequality, and $\Delta(x, y) = 0$ if and only if $x = y$. $\square$

The distance Δ is a real number; it leads to the definition of an *amplitude-code* in the ring R which characteristics (n, k, d) are real numbers. Informally, Δ corresponds to the maximal amount of information that differs between the two words.

DEFINITION 2.2. *Let $\Pi = \prod_{i=1}^m P_i$, where $P_1, \dots, P_m$ are pairwise relatively prime. Let $n = \log_2 \nu(\Pi)$, $k = \log_2 \kappa$. A (n, k, d) -amplitude code is a subset $\mathcal{C} = \{x \in R \mid \nu(x) < \kappa\}$ of the set $\mathcal{A} = \{x \in R \mid \nu(x) < \nu(\Pi)\}$ such that the minimal distance between two elements of $\mathcal{C}$ is d .*

PROPOSITION 2.3. *For any (n, k, d) -amplitude code,*

$$d > n - k - 1$$

PROOF. Let $X, Y \in \mathcal{C}$ and $X \neq Y$. Let $E = Y - X$ and I be the subset of $\{1, \dots, m\}$ such that $X \neq Y \bmod P_i$ if and only if $i \in I$. Let $\Pi_F = \prod_{i \in I} P_i$ and $\Pi_T = \Pi / \Pi_F$. $\Delta(X, Y) = \log_2 \nu(\Pi_F)$

Now E is a multiple of Π_T , as $E = 0 \pmod{P_i}$ for any $i \notin I$. Therefore

$$\nu(E) \geq \nu(\Pi)/\nu(\Pi_F)$$

On the other hand $\nu(E) \leq \nu(X) + \nu(Y) < 2\kappa$. Consequently $\nu(\Pi)/\nu(\Pi_F) < 2\kappa$ and then $d > n - k - 1$. $\square$

COROLLARY 2.4. *When the Euclidean function ν satisfies the following identity: $\nu(a + b) = \max(\nu(a), \nu(b))$ for all $a, b \in R$, then a tighter bound on the minimal distance can be achieved:*

$$d > n - k.$$

Moreover, in the case of the ring $R = K[x]$, where K is a field, choosing $\nu(a) = 2^{\deg(a)}$ leads to the equivalent of the Singleton bound $d \geq n - k + 1$.

PROOF. In that case $\nu(E) = \max(\nu(X), \nu(Y)) < \kappa$. It follows $\nu(\Pi)/\nu(\Pi_F) < \kappa$ and then $d > n - k$. When $R = K[x]$ and $\nu(a) = 2^{\deg(a)}$, the characteristics n, k, d are integers, therefore $d \geq n - k + 1$. $\square$

In the setting of coding theory, the minimal distance gives a bound on the maximal number of errors that can be corrected: $t = \lfloor \frac{d-1}{2} \rfloor$. Here, it corresponds to the maximal amplitude that can be corrected. More precisely, let $\Pi_F = \prod_{i \in I} P_i$ be the impact, and $\nu(\Pi_F)$ its amplitude. Then a code will be *amplitude- τ corrector* if it can correct errors of impact Π_F , such that $\nu(\Pi_F) \leq \tau$.

Hence, we could aim at correcting impact of amplitude τ such that $\log_2 \tau = \frac{d-1}{2} = \frac{n-k-2}{2}$. This approach prevents us from reaching the optimal correction rate of [9] but remains close to it when the valuations $\nu(P_1), \dots, \nu(P_n)$ remain of the same order of magnitude.

2.2 The bounded impact amplitude algorithm

The Euclidean function and metric defined in the previous section allow to unify both algorithms from [16] for integers and [19, 5] for polynomials. In the following, the Euclidean ring R can be either $\mathbb{Z}$ or $K[X]$, where K is a field. We present the decoding algorithm based on the usual extended Euclidean algorithm. It allows to correct up to the largest impact amplitude with respect to the bound on the minimal distance (property 2.3).

THEOREM 2.5. *Algorithm 1 decodes any corrupted message Y originating from a codeword X affected by an error impact Π_F such that $4\nu(X)\nu(\Pi_F)^2 \leq \nu(\Pi)$ and $\nu(\Pi_F) \leq \tau$.*

PROOF. In the last iteration of the loop, the invariant relation in the extended Euclidean algorithm is written

$$\alpha_{i+1}\Pi - \beta_{i+1}Y = r_{i+1}$$

Developing the noisy message $Y = E + X$ gives:

$$\alpha_{i+1}\Pi - \beta_{i+1}E = r_{i+1} + \beta_{i+1}X.$$

Now $\gcd(\Pi, E) = \Pi_T$, therefore there is a $Z \in R$ such that

$$Z\Pi_T = r_{i+1} + \beta_{i+1}X.$$

We now will prove that $Z = 0$ which implies that $X = -\frac{r_{i+1}}{\beta_{i+1}}$ upon termination of the algorithm.

A usual result [6, Lemma 5.15] in the extended Euclidean algorithm over R states that $\nu(\beta_{i+1}) \leq \frac{\nu(\Pi)}{\nu(r_i)}$ (the inequality is strict in $\mathbb{Z}$ and becomes an equality in $K[X]$). The

Algorithm 1: Amplitude based decoder over R

Data: $\Pi \in R$: the product of the moduli
Data: $Y \in R$: the possibly erroneous message
Data: $\tau \in \mathbb{R}_+ \mid \tau < \frac{\nu(\Pi)}{2}$: a bound on the amplitude of the maximal impact of an error to correct
Result: $X \in R$: the corrected message satisfying $\nu(X)4\tau^2 \leq \nu(\Pi)$

begin
 $\alpha_0 = 1, \beta_0 = 0, r_0 = \Pi$;
 $\alpha_1 = 0, \beta_1 = 1, r_1 = Y$;
 $i = 1$;
while $(\nu(r_i) > \nu(\Pi)/2\tau)$ **do**
 Let $r_{i-1} = q_i r_i + r_{i+1}$ be the Euclidean division of r_{i-1} by r_i ;
 $\alpha_{i+1} = \alpha_{i-1} - q_i \alpha_i$;
 $\beta_{i+1} = \beta_{i-1} - q_i \beta_i$;
 $i = i + 1$;
return $X = -\frac{r_i}{\beta_i}$
end

condition of termination of the **while** loop writes

$$\nu(r_{i+1}) \leq \frac{\nu(\Pi)}{2\tau} < \nu(r_i)$$

Hence, $\nu(\beta_{i+1}) < 2\tau$, and we have

$$\begin{aligned} \nu(Z \cdot \Pi_T) &\leq \nu(r_{i+1}) + \nu(X)\nu(\beta_{i+1}) \\ &< \nu(r_{i+1}) + 2\tau\nu(X) \\ &< \nu(r_{i+1}) + 2\tau \frac{\nu(\Pi)}{4\tau^2} \\ &< \frac{\nu(\Pi)}{2\tau} + \frac{\nu(\Pi)}{2\tau} \end{aligned}$$

Now, by definition $\nu(\Pi_T) \geq \nu(\Pi)/\tau$. Finally the only possible value for Z is 0. $\square$

COROLLARY 2.6. *When the Euclidean function ν satisfies the identity $\nu(a+b) = \max(\nu(a), \nu(b))$ for all $a, b \in R$, then a tighter bound for the error correction bound can be achieved: $\nu(X)2\tau^2 \leq \nu(\Pi)$. This is the case for the ring $R = K[x]$, where K is a field, and $\nu(a) = 2^{\deg(a)}$.*

In this situation again, the algorithm corrects up to the largest error amplitude with respect to the bound on the minimal distance given in corollary 2.4.

PROOF.

$$\nu(Z\Pi_T) \leq \max(\nu(r_{i+1}), \nu(X)\nu(\beta_{i+1}))$$

Now $\nu(r_{i+1}) \leq \frac{\nu(\Pi)}{2\tau}$ and

$$\begin{aligned} \nu(X)\nu(\beta_{i+1}) &< 2\tau \frac{\nu(\Pi)}{2\tau^2} \\ &< \frac{\nu(\Pi)}{\tau} \end{aligned}$$

Therefore

$$\nu(Z\Pi_T) < \frac{\nu(\Pi)}{\tau}.$$

Since $\nu(\Pi_T) \geq \nu(\Pi)/\tau$ we have $Z = 0$. $\square$

The next proposition states the complexity of algorithm 1.

PROPOSITION 2.7. *The complexity of algorithm 1 (arithmetic complexity over $K[X]$ or bit complexity over $\mathbb{Z}$) is $\mathcal{O}(tn)$, where $t \leq \log_2 \tau$ is the size of the error.*

PROOF. From the analysis of the classical Extended Euclidean algorithm [12, §4.5.3] applied to two integers or polynomials of size n the number of iterations to reach a remainder r_i with size λ is less than $2(n - \lambda) = \mathcal{O}(t)$. The total complexity is $\mathcal{O}(nt)$ using classical arithmetic. $\square$

Fast algorithm using HGCD.

The complexity can be improved based on the use of a fast truncated gcd algorithm. Indeed, the stopping condition ($\nu(r_i) > \nu(\Pi)/2\tau$) is related to the valuation of the remainder r_i . This valuation is derived from the size of r_i (degree for a polynomial, $\log_2$ for an integer) without requiring a computation of the exact value of r_i . Thus, a divide-and-conquer algorithm, such as HGCD [6, §11], [20], can be used to compute the Euclidean sequence of quotients but not the exact remainders, until the modified stopping condition (based on the size) is reached: then only, the corresponding remainder is computed. The complexity of this algorithm becomes $\mathcal{O}(\log t \cdot M(n))$ where $M(n)$ is the cost of the multiplication of two elements of size n in R .

2.3 Comparison with previous decoding algorithms

The decoding algorithm 1 allows to correct a bounded amplitude of the error impact in a (n, k, d) -amplitude code defined over $\mathbb{Z}$ or $K[X]$ equipped with a sub-additive and multiplicative euclidean function. As it has been mentioned in previous sections algorithm 1 can be seen as a generalization of the algorithm of Mandelbaum [16] over the set of integers or a generalisation of the algorithms of Shiozaki [19] and Gao [5] over the set of polynomials.

Furthermore, when $R = K[X]$ and the P_i 's all have degree one, then algorithm 1 corresponds exactly to that of Gao [5] presented as an alternative decoder for Reed-Solomon codes. The bounds and error correction capacity are also the same.

Apart from the special case of Reed-Solomon codes, our representation makes a significant difference in the correction capacity. Algorithm 1 is only correcting an amplitude of error impact: this does not mean anything on the number of erroneous residues that can actually be corrected. Therefore, two single errors affecting two different moduli P_i and P_j such that $\nu(P_i) < \nu(P_j)$ don't have the same weight in the error correction. This fact fits naturally in our setting but must be masked in the standard representation (any moduli should have the same weight in the correction rate of the code). Hence the correction algorithms usually have to be "patched" to avoid the problem and then match the optimal correction capacity: Mandelbaum algorithm uses a perturbation technique and requires the moduli to stay relatively close, equivalently the algorithm of Shiozaki require the moduli to have same degree. On the other hand, Gol-dreich, Ron and Sudan [7] modify the correction capacity in order to make it dependent to the difference of amplitude of the moduli. Finally Guruswami & Al. [9] use weights on the residues to express this distortion.

3. OUTPUT SENSITIVE DECODING AND EARLY TERMINATION

The Chinese remainder algorithm requires a bound on the result to be reconstructed which in turn determines how many modular computations need to be done. In practice the bound might be pessimistic, and a smaller number of modular computations are sufficient for the reconstruction. We show in sections 3.1 and 3.2 how to take advantage of this unnecessary redundancy to increase the error correction capacity, generalizing algorithm 1 into an adaptive algorithm.

Now this approach still assumes that the total number of modular computations is fixed. Section 3.4 proposes an early termination framework, that allows both adaptive correction rate and output sensitive number of modular computations.

3.1 A first adaptive heuristic

We assume here that no information on the bound κ nor any bound on the maximal error amplitude τ is known, but only the values of the product Π , and the message Y . The general approach is to define a termination criterion based on the current state of the execution of the Extended Euclidean algorithm.

This first criterion is adapted from a step in the algorithm by Mandelbaum [16]. The idea is to check that β_{i+1} divides Π . Indeed, since the error $E = Y - X$ satisfies: $\alpha_{i+1}\Pi - \beta_{i+1}E = 0$, we have $E = \alpha_{i+1} \frac{\Pi}{\beta_{i+1}}$, which implies that β_{i+1} divides $\Pi\alpha_{i+1}$. More precisely, lemma 3.1 shows that β_{i+1} actually divides Π , yet is a multiple of the impact Π_F .

LEMMA 3.1. *Suppose that $\nu(\Pi) \geq \nu(X)4\nu(\Pi_F)^2$. Let i be the iteration where the relation*

$$\nu(r_{i+1}) \leq \frac{\nu(\Pi)}{2\nu(\Pi_F)} < \nu(r_i)$$

holds. Then Π_F divides β_{i+1} and $\nu(\beta_{i+1}) = \nu(\Pi_F)$.

PROOF. First, $\nu(\beta_{i+1}) \leq \frac{\nu(\Pi)}{\nu(r_i)} < 2\nu(\Pi_F)$. Now, $\alpha_{i+1}\Pi - \beta_{i+1}E = 0$ (applying the proof of theorem 2.5 with $\tau = \Pi_F$), hence Π_F divides $E\beta_{i+1}$. Since E and Π_F are relatively prime, we deduce that Π_F divides β_{i+1} : there exist $\gamma \in R$ such that $\beta_{i+1} = \gamma\Pi_F$.

Therefore $\nu(\Pi_F) \leq \nu(\beta_{i+1}) < 2\nu(\Pi_F)$ which implies that $\nu(\gamma) = 1$. $\square$

Lemma 3.1 implies two necessary conditions for X to be a valid codeword: β_j divides Π and

$$\nu(X) \leq \frac{\nu(\Pi)}{4\nu(\beta_j)^2},$$

at some iteration j of the Extended Euclidean algorithm applied to Π and Y .

Algorithm 2 summarizes the adaptive algorithm based on divisibility check.

Since no information is known about the maximal size of the codewords κ or the maximal error correction capacity τ the algorithm returns a list of candidates that all satisfy a condition of the type $\nu(X)4\tau^2 \leq \nu(\Pi)$, for varying values of τ . In order to discriminate the right solution, one needs to check a postcondition on the result, as will be described in section 3.4.

Algorithm 2: Adaptive decoding by divisibility check

Data: $\Pi \in R$: the product of the moduli
Data: $Y \in R$: the possibly erroneous message
Result: C : a list of possible candidates X_i satisfying:
 $\nu(X_i)4\tau^2 \leq \nu(\Pi)$ if an error of amplitude τ occurred

```

begin
   $\alpha_0 = 1, \beta_0 = 0, r_0 = \Pi$ 
   $\alpha_1 = 0, \beta_1 = 1, r_1 = Y$ 
   $i = 1, C = \{\}$ 
  while  $\nu(r_i) > 0$  do
    Let  $r_{i-1} = q_i r_i + r_{i+1}$  be the Euclidean division
    of  $r_{i-1}$  by  $r_i$ 
     $\alpha_{i+1} = \alpha_{i-1} - q_i \alpha_i$ 
     $\beta_{i+1} = \beta_{i-1} - q_i \beta_i$ 
    if  $\beta_{i+1}$  divides  $\Pi$  then
       $X = -\frac{r_{i+1}}{\beta_{i+1}}$ 
      if  $\nu(X) \leq \frac{\nu(\Pi)}{4\nu(\beta_{i+1})^2}$  then
        Push  $X$  in  $C$ 
       $i = i + 1$ 
    return  $C$ 
end

```

3.2 Detecting a gap

An alternative termination criterion is to consider the size of the quotients in the Euclidean algorithm. We consider a modified bound on the amplitude of Π :

$$\nu(\Pi) \geq 4\nu(X)\nu(\Pi_F)^2 2^g,$$

where g is an arbitrarily chosen positive integer that we will refer to as the gap. Introducing this gap, somehow limits the maximal size of error $\nu(\Pi_F)$ that can be corrected, or equivalently, limits the maximal size of X . Thus, the product Π_T of the correct moduli includes a larger amount of redundancy, characterized by $\nu(\Pi_T) \geq 4\nu(X)\nu(\Pi_F)^2 2^g$. This will be used to detect the termination of the decoding: lemma 3.2 states that at the last step i of the extended Euclidean algorithm, the size of the quotient q_i is necessarily larger than the gap g .

LEMMA 3.2. *Suppose that $\nu(\Pi) = \nu(X)4\nu(\Pi_F)^2 2^g$ with $g \in \mathbb{Z}_+$. Let i be the iteration where the relation*

$$\nu(r_{i+1}) \leq \frac{\nu(\Pi)}{2\nu(\Pi_F)} < \nu(r_i)$$

holds. Then $\nu(q_{i+1}) \geq 2^g$.

PROOF. First, $\nu(\beta_{i+1}) \leq \frac{\nu(\Pi)}{\nu(r_i)} < 2\nu(\Pi_F)$. Now, $\alpha_{i+1}\Pi - \beta_{i+1}E = 0$ (applying the proof of theorem 2.5 with $\tau = \Pi_F$), hence $r_{i+1} = \beta_{i+1}X$ and then $\nu(r_{i+1}) = \nu(\beta_{i+1})\nu(X)$. Consequently $\nu(r_{i+1}) < 2\nu(\Pi_F)\nu(X)$.

On another hand, $\nu(r_i) > \frac{\nu(\Pi)}{2\nu(\Pi_F)} = 2\nu(X)\nu(\Pi_F)^2 2^g$. Thus $\frac{\nu(r_i)}{\nu(r_{i+1})} > 2^g$. Now, $\nu(r_i) \leq \nu(r_{i+2}) + \nu(q_{i+1})\nu(r_{i+1})$ implies $\nu(r_i) < r_{i+1}(1 + \nu(q_{i+1}))$. Finally

$$2^g < \frac{\nu(r_i)}{\nu(r_{i+1})} \leq 1 + \nu(q_{i+1})$$

which completes the proof. $\square$

Lemma 3.2 states a necessary condition for $X = -\frac{r_j}{\beta_j}$ to be a valid codeword at the iteration j of the Extended Euclidean algorithm applied to Π and Y : $\nu(q_j) \geq 2^g$. This

property is used in algorithm 3 : only steps corresponding to a quotient with valuation larger than the gap are considered for the recovery of X . The main interest is to decrease the complexity, since the average amplitude for the quotient in the Euclidean algorithm is small: less than 3 for the integers in average [12].

Algorithm 3: Adaptive algorithm, by detection of a gap

Data: $\Pi \in R$: the product of the moduli
Data: $g \in \mathbb{Z}_+$: the size of the gap
Data: $Y \in R$: the possibly erroneous message
Result: C : a list of possible candidates X_i satisfying:
 $\nu(X_i)4\tau^2 2^g = \nu(\Pi)$ if an error of amplitude τ occurred

```

begin
   $\alpha_0 = 1, \beta_0 = 0, r_0 = \Pi$ 
   $\alpha_1 = 0, \beta_1 = 1, r_1 = Y$ 
   $i = 1, C = \{\}$ 
  while  $\nu(r_i) > 0$  do
    Let  $r_{i-1} = q_i r_i + r_{i+1}$  be the Euclidean division
    of  $r_{i-1}$  by  $r_i$ 
    if  $\nu(q_i) \geq 2^g$  then
      if  $\beta_i$  divides  $\Pi$  then
         $X = -\frac{r_i}{\beta_i}$ 
        if  $\nu(X) \leq \frac{\nu(\Pi)}{4\nu(\beta_i)^2}$  then
          Push  $X$  in  $C$ 
         $\alpha_{i+1} = \alpha_{i-1} - q_i \alpha_i$ 
         $\beta_{i+1} = \beta_{i-1} - q_i \beta_i$ 
         $i = i + 1$ 
      return  $C$ 
    end
  end

```

Similarly to algorithm 2, the algorithm based on the gap's criterion allows to keep the valuations of X and Π_F unknown (as far as the relation $\nu(X)4\nu(\Pi_F)^2 2^g = \nu(\Pi)$ is verified). Even though the presence of a gap seems to decrease the overall capacity of the code compared to algorithm 2, it allows to restrain the number of acceptable candidate: we will see in section 3.3 (table 1) that a relatively small size of the gap (i.e. in our experiments in $\mathbb{Z}$, a small number of extra redundant bits in comparison to the size of one moduli) leads to a list of only one or two valid candidates and reduces the number of divisibility checks as much.

Again here, the removal of the invalid remaining candidates in the list C needs to be done, for instance using an external certifier (see section 3.4).

Fast algorithm using HGCD.

Interestingly, algorithm 3 allows to use a fast Extended Euclidean algorithm instead of the standard algorithm. Indeed, when the algorithm *HGCD* [6, §11], [20], improves the complexity by an order of magnitude, it still constructs the whole list of quotients of the classic Extended Euclidean Algorithm. Algorithm 3, with a termination criterion based on the size of the quotient, can then be easily upgraded in a fast version.

On the contrary, algorithm 2 needs to check every remainder of the Extended Euclidean Algorithm and then won't allow to use a fast version.

3.3 Experimental comparison

We implemented these algorithms for $R = \mathbb{Z}$ in C using the GMP library [8] for the multi-precision integer arithmetic. In each experiment, Π is the product of the first m prime numbers of 21 bits, and κ is the product of the first $\ell < m$ of them.

In figure 2 and 1 we compare the computation time of

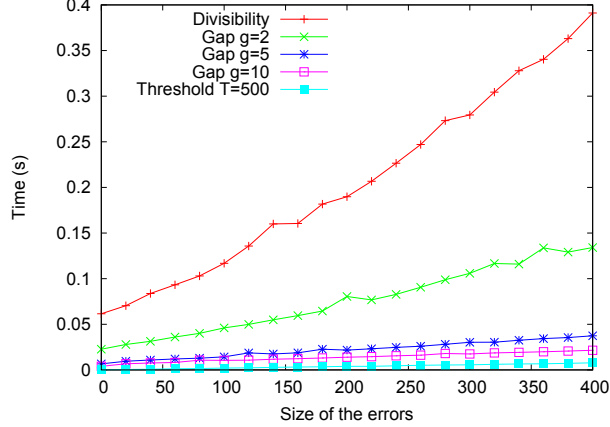


Figure 1: Comparison of the variants of the decoder for $n \approx 26\,016$ ($m = 1300$ moduli of 20 bits), $\kappa \approx 6001$ (300 moduli) and $\tau \approx 10007$ (about 500 moduli).

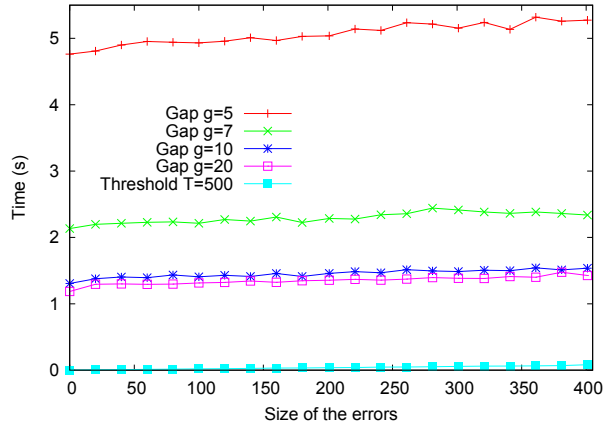


Figure 2: Comparison of the variants of the decoder for $n \approx 200\,917$ ($m = 10000$ moduli of 20 bits), $\kappa \approx 17\,0667$ (8500 moduli) and $\tau \approx 10498$ (500 moduli).

the static algorithm 1 (Threshold), and the adaptative algorithms 2 (Divisibility) and algorithm 3 (Gap detection), for three parameters of the gap: $g = 2, 5, 10$.

First the best computation time is always achieved by algorithm 1, where the parameters are known, since it requires no additional computation for the termination. Now if no information is known on κ , the first adaptive algorithm, based on divisibility checks, is slowed down by these expensive tests, performed at each iteration of the Euclidean algorithm. The gap detection of algorithm 3 is much cheaper

(a simple evaluation of the size of the current quotient q_i). Now the gap can be set arbitrarily: if it is large, it will reduce the maximal error capacity, for a given n and X , and if it is too small, it will reveal several false positives, that need to be discarded by a test of divisibility. For the extremely small value of $g = 2$, the computation time is already significantly reduced, compared to algorithm 2. Then changing it to $g = 5$ bits, still drastically reduce the computation time, as it reduces the number of false positives. For $g = 10$ (half the size of a modulo), the computation time almost matches that of the static algorithm 1.

Now, since no information on the size of the message is known, the gap algorithm has to keep going in the Extended Euclidean algorithm down to the end. Thus if the amplitude of X is relatively large compared to Π , this will make significant useless amount of work. Figure 1 shows this phenomenon: as the gap g increases, the computation time reduces down to a limit, that roughly corresponds to the time of executing the Extended Euclidean algorithm to the end. The static algorithm, takes advantage of the a priori knowledge of the threshold τ to terminate earlier. This motivates the introduction of an external certification algorithm, that will test each candidate produced by the decoder *on the fly*, against a few independent additional modular residues, and stop the decoder whenever a certification succeeded. This will be described in section 3.4.

Error size	10	50	100	200	500	1000
$g = 2$	$1/446$	$1/765$	$1/1118$	$2/1183$	$2/4165$	$1/7907$
$g = 3$	$1/244$	$1/414$	$1/576$	$2/1002$	$2/2164$	$1/4117$
$g = 5$	$1/53$	$1/97$	$1/153$	$2/262$	$1/575$	$1/1106$
$g = 10$	$1/1$	$1/3$	$1/9$	$1/14$	$1/26$	$1/35$
$g = 20$	$1/1$	$1/1$	$1/1$	$1/1$	$1/1$	$1/1$

Table 1: Number of candidates in the gap algorithm: c/d means that d candidates appeared with a gap larger than g , and c out of them passed the divisibility check. $n \approx 6001$ (3000 moduli), $\kappa \approx 201$ (100 moduli).

However, the amount of candidates that remain after the divisibility check is very small, and in most cases there is only one of them: the correct result. Table 1 displays the fraction of candidates passing the divisibility check over the total number of candidates passing the gap condition for different values of g and error size.

3.4 A framework for early termination

Early termination

In the previous section, the number m of modular computations was fixed, and only the correction capacity was made output-sensitive. In order to limit the number of modular computations, early termination Chinese remaindering is commonly used [11, 3]: an increasing number of modular residues are computed until the reconstructed value in R stabilizes. In the context of parallel computing, a chunk of modular computations will be done in parallel at each step and if the stabilization condition is not met, then another chunk will be computed. Following [1] the cardinality u_i of each chunk needs to follow an amortized function f in order to guarantee a minimal work overhead. They are defined as

follows $u_1 = C$ and $u_{i+1} = f(u_i)$. The function f could be e.g. $f(x) = x/\log x$ or $f(x) = x^{1-\epsilon}$, with $1 \gg \epsilon > 0$.

Certifier

The decoder (algorithm 3) only returns a list of candidates $\langle X^{(1)}, X^{(2)}, \dots \rangle$, since the parameters of the code are unknown. Therefore a certifier is needed, in order to withdraw the invalid candidates. This can be implemented by a simple algorithm, maintaining a list of results $(r_i \bmod P_i)$ of modular computations, independent from the one used in the decoder. The certification consists in testing if $X^{(j)} = r_i \bmod P_i$ and return the residue that succeeds every tests. By choosing an appropriate number of r_i 's, any probability of success can be guaranteed (refer to [11] for a detailed analysis).

Note that this certifier naturally plays the role of checking the stabilization, as needed for the early termination.

Chinese remaindering lifter

The residues computed on each distributed node must be lifted using the Chinese remainder theorem to form the message Y . This operation could either be done on a safe resource running the *Decoder*, but this can be a significant work load. Another strategy is to have every worker do the reconstruction, and then select the appropriate one by a majority vote, thus preventing a possible corruption by a corrupted node.

A framework for global computing

We now describe a framework for an output-sensitive distributed computation using the components described previously. Following [17], the computing environment is partitioned in two parts: U and R . R is a closed set of reliable interconnected computing resources; R is assumed reliable and trustfully, thus it provides a limited computation power π_R . On the other hand, U includes a large number of resources that operate in an unbounded environment, such as a peer to peer computation environment, and provides a very large computation power π_U . Such a large scale parallel system is well suited to perform independent computations, such as modular computations. However, since U is open, computations performed on U cannot be considered as fully reliable which motivates the introduction of a fault tolerant reconstruction system.

Figure 3 summarizes the framework of such a system. A program called *Master* forks the parallel computations following the amortized function f . Then the value R is reconstructed by the *Lifter* and the *Decoder* (algorithm 3) lists all codewords X that are candidates. The *Certifier* then discards the invalid candidates. If no candidate is left, then *Master* forks the next chunk of modular computations. Otherwise, the probabilistically certified result is returned.

4. CONCLUSION

We presented adaptations of the well studied unique decoding algorithm for redundant residue systems, making its error correction capacity adaptive in the effective amount of redundancy. In this process, we cut free from the usual description of the code in terms of residues, but instead described it directly over the ring, using the euclidean function, and a specific metric for the distance between two elements. The (n, k, d) parameters of the code now refer to

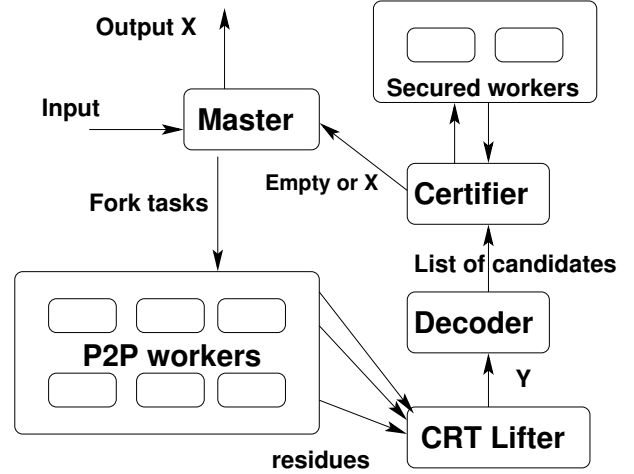


Figure 3: Framework for an output-sensitive fault tolerant distributed computation

the corresponding amplitudes in the ring and we provided lower bounds on the minimal distance, matching the Singleton bound in the case of a ring of polynomials over a field.

In this more general framework, the moduli no longer need to be sorted, and the correction capacity is more tightly bounded to the total amount of redundancy available. Algorithm 1 is presented over an Euclidean ring, but we could only prove its validity and that it achieves the maximal error correction capacity, given by the previous bounds in the case where $R = \mathbb{Z}$ or $R = K[x]$. In these proofs, the argument that $\nu(\beta_{i+1}) \leq \frac{\nu(\Pi)}{\nu(r_i)}$ was repeatedly needed. So far, we were not able to prove it in the general context of an Euclidean ring, with a multiplicative and sub-additive valuation ν , and we are not aware of any such result in the literature. It can be reduced to showing that

$$\nu(\Pi) = \nu(\beta_{i+1}r_i - \beta_i r_{i+1}) \geq \nu(\beta_{i+1}r_i).$$

It holds over $\mathbb{Z}$ and $K[X]$, but for two different reasons: over $\mathbb{Z}$ because the β_i 's alternate in sign, and over $K[X]$ by an argument on the degrees. This raises the question to know what the least requirements on the ring or the valuation ν are, for this to be true.

We then introduced two termination criterions to form an adaptive decoding algorithm that performs very efficiently in practice: allowing a very small amount of extra redundancy (less than half of the amplitude of one modulo), it tends to be as fast as the static algorithm especially if it can be interrupted by an external certifier.

A further analysis on the average distribution of the false positives appearing in the gap algorithm, as a function of the gap, would help understand why this parameter only needs to be set to extremely small values in practice.

5. REFERENCES

- [1] J. Bernard, J.-L. Roch, and D. Traore. Processor-oblivious parallel stream computations. In *16th Euromicro International Conference on Parallel, Distributed and network-based Processing*, Toulouse, France, Feb 2007.

- [2] G. Bosilca, R. Delmas, J. Dongarra, and J. Langou. Algorithm-based fault tolerance applied to high performance computing. *Journal of Parallel and Distributed Computing*, 69(4):410 – 416, 2009.
- [3] J.-G. Dumas, C. Pernet, and Z. Wan. Efficient computation of the characteristic polynomial. In *ISSAC '05: Proceedings of the 2005 international symposium on Symbolic and algebraic computation*, pages 140–147, New York, NY, USA, 2005. ACM.
- [4] I. Z. Emiris. A complete implementation for computing general dimensional convex hulls. *International Journal of Computational Geometry and Applications*, 8(2):223–253, Apr. 1998.
- [5] S. Gao. A new algorithm for decoding reed-solomon codes. In *in Communications, Information and Network Security, V.Bhargava, H.V.Poor, V.Tarokh, and S.Yoon*, pages 55–68. Kluwer, 2002.
- [6] J. v. Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 1999.
- [7] O. Goldreich, D. Ron, and M. Sudan. Chinese remaindering with errors. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 225–234, New York, NY, USA, 1999. ACM.
- [8] T. Granlund. *The GNU multiple precision arithmetic library*, 2010. Version 4.3.2, <http://gmplib.org/manual-4.3.2/>.
- [9] V. Guruswami, A. Sahai, and M. Sudan. "soft-decision" decoding of chinese remainder codes. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 159, Washington, DC, USA, 2000. IEEE Computer Society.
- [10] K.-H. Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Trans. Computers*, 33(6):518–528, 1984.
- [11] E. Kaltofen. An output-sensitive variant of the baby steps/giant steps determinant algorithm. In *ISSAC '02: Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 138–144, New York, NY, USA, 2002. ACM.
- [12] D. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Reading, MA., 1981.
- [13] H. Krishna, K.-Y. Lin, and J.-D. Sun. A coding theory approach to error control in redundant residue number systems. i. theory and single error correction. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 39(1):8–17, Jan 1992.
- [14] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.
- [15] T. Liew, L.-L. Yang, and L. Hanzo. Soft-decision redundant residue number system based error correction coding. In *VTC'99 (Fall)*, pages 2546–2550, September 1999.
- [16] D. Mandelbaum. On a class of arithmetic codes and a decoding algorithm (corresp.). *Information Theory, IEEE Transactions on*, 22(1):85–88, Jan 1976.
- [17] J.-L. Roch and S. Varrette. Probabilistic certification of divide & conquer algorithms on global computing platforms: application to fault-tolerant exact matrix-vector product. In *PASCO '07: Proceedings of the 2007 international workshop on Parallel symbolic computation*, pages 88–92, New York, NY, USA, 2007. ACM.
- [18] F. B. Schneider. Byzantine generals in action: implementing fail-stop processors. *ACM Trans. Comput. Syst.*, 2(2):145–154, 1984.
- [19] A. Shiozaki. Decoding of redundant residue polynomial codes using euclid's algorithm. *Information Theory, IEEE Transactions on*, 34(5):1351–1354, Sep 1988.
- [20] K. Thull and C. Yap. A unified approach to HGCD algorithms for polynomials and integers, 1990. Manuscript. Available from <http://cs.nyu.edu/cs/-faculty/yap/allpapers.html/>.
- [21] R. Watson and C. Hastings. Self-checked computation using residue arithmetic. *Proceedings of the IEEE*, 54(12):1920–1931, Dec. 1966.

A Strassen-Like Matrix Multiplication Suited for Squaring and Higher Power Computation

Marco Bodrato

Centro Interdipartimentale “Vito Volterra”
Universit degli Studi di Roma “Tor Vergata”
Via Columbia 2, 00133 Roma (Italy)
bodrato@volterra.uniroma2.it

ABSTRACT

Strassen’s method is not the asymptotically fastest known matrix multiplication algorithm, but it is the most widely used for large matrices. Since his manuscript was published, a number of variants have been proposed with different addition complexities. Here we describe a new one. The new variant is at least as good as those already known for simple matrix multiplication, but can save operations either for chain products or for squaring. Moreover it can be proved optimal for these tasks. The largest saving is shown for n^{th} -power computation, in this scenario the additive complexity can be halved, with respect to original Strassen’s.

Categories and Subject Descriptors

F.2.1 [Analysis of algorithms and program complexity]: Numerical algorithms and problems—*Computations on matrices*; G.1.3 [Numerical Analysis]: Numerical Linear Algebra; G.2.3 [Discrete mathematics]: Applications; I.1.2 [Computing methodologies]: Algorithms—*Algebraic algorithms*

General Terms

Algorithms, Performance, Theory

Keywords

Polynomial matrix, exponentiation, optimal squaring, fast multiplication, recursive algorithm.

1. INTRODUCTION

When this work started, the main goal was to find a way to speed up the evaluation of a polynomial on a matrix. We started from the first monomial that needs some non-trivial computation: x^2 .

To speed up this simple operation, we analysed Strassen’s method [19] for matrix multiplication: it is the most widely

used, even if is not the asymptotically fastest known. Every possible Strassen-like algorithm uses the same number of multiplications: they differ on the number of additions and subtractions. Winograd’s variant [21] requires the minimal use of linear operations: 15 for a 2×2 matrix. None of the algorithms proposed so far did try to minimise operations for squaring.

The new sequence we propose in §2.2, is basically equivalent to Winograd’s variant, with a simple additional property: symmetry. All the results in this paper derive from this symmetry, directly or as a side effect.

The direct result is optimality for squaring, because we can exploit the obvious symmetry of multiplying a single operand by itself. The main side effect is a reduction of the number of linear combinations needed either for the chain product of three or more matrices, or required by n^{th} -power computation or even by more general polynomials on matrices, the initial goal.

While matrix product and Strassen’s algorithm can be applied to rectangular matrices, this paper focuses on square matrices, because only for this kind of matrices do squaring and higher power computation make sense. Nevertheless the new proposed sequence can be extended to any multiplication using standard techniques [6, 10].

To obtain formulas valid on every ring we started searching combinations valid for boolean matrices, on $\mathbb{F}_2$. Then we extended the sequences found this way to characteristic 0, testing all possible liftings of the same formulas where $1 \in \mathbb{F}_2$ was lifted to $\pm 1 \in \mathbb{Z}$. Since the obtained algorithms require only additions, subtractions and (non-commutative) multiplications, they will work on any ring. In particular they will work on the algebra of matrices, and can be used recursively.

Applications

Strassen’s and Strassen-like methods for matrix multiplication are considered numerically unstable [2]; although some corrections are possible, the main application for a fast matrix multiplication algorithm is that of matrices on exact rings or algebras. We mean matrices over finite fields, integers, rationals, polynomials and so on.

Many algebraic algorithms in graph theory work by building a matrix somehow representing the graph, then computing some power of it, or repeatedly squaring it [9]. The best-known result is: check elements on the diagonal of the k^{th} power of the adjacency matrix to count closed k -walks in a graph. Adjacency matrices of graphs usually have boolean or integer entries, but also different definitions of the ad-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

jacency [18] need an algebra where exact computations are possible and Strassen-like methods can be applied.

Another direct application of matrix exponentiation can be cryptography [17, 15]. Moreover the reduced number of operations for chain products can be used in a lot of different contexts, e.g. for number theoretical algorithms where small 2×2 matrices with huge integers are multiplied, like the half-GCD [14] or fast Jacobi symbol computation [4].

2. MATRIX PRODUCT

Given two $d \times d$ matrices A, B , computation of the product $C = AB$, with a naïve implementation, directly applying the definition $C_{ij} = \sum_k A_{ik} B_{kj}$, requires d^3 multiplications and $d^3 - d^2$ additions.

In particular we will study the 2×2 case, so we will have:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \quad (1)$$

In the Appendix §Q only, we will briefly discuss the commutative case. For the rest of the paper we will consider the non-commutative scenario, because for big $2k \times 2k$ matrices the A_{ij}, B_{ij} are not ring elements, but $k \times k$ matrices themselves.

2.1 Strassen-like Algorithms

The naïve algorithm requires $\Theta(d^3)$ operations, but asymptotically faster algorithms exist. The first one, proposed in 1969 by Strassen [19], has the lower complexity $\Theta(d^{\log_2 7}) = O(d^{2.8074})$. This is not the “fastest” known, but it’s the most widely used, because asymptotically faster algorithms are efficient for very huge matrices only.

The basic idea consists in an optimisation for the product of 2×2 matrices, requiring only 7 instead of 8 multiplications. When used recursively it gives the product of $2^k \times 2^k$ matrices using 7^k ring multiplications, instead of $2^{3k} = 8^k$.

When applied to 2×2 split matrices, Strassen method requires also 18 linear operations (additions/subtractions) on the sub-matrices.

2.2 New Proposed Sequence

We propose here a new variant, requiring the minimum number of operations, and with an additional property: it is optimal for squaring too and can save operations for chain (three or more matrices) products. We start by describing it for the simple product.

At first four linear pre-combinations are required for each one of the two operands:

$$\begin{aligned} (2.s) \quad & \begin{cases} S_1 = A_{22} + A_{12} \\ S_2 = A_{22} - A_{21} \\ S_3 = S_2 + A_{12} = A_{22} - A_{21} + A_{12} \\ S_4 = S_3 - A_{11} = A_{22} - A_{21} + A_{12} - A_{11} \end{cases} \\ (2.t) \quad & \begin{cases} T_1 = B_{22} + B_{12} \\ T_2 = B_{22} - B_{21} \\ T_3 = T_2 + B_{12} \\ T_4 = T_3 - B_{11} \end{cases} \end{aligned} \quad (2)$$

then the seven products and the final post-combinations:

$$(3.p) \quad \begin{cases} P_1 = S_1 T_1 \\ P_2 = S_2 T_2 \\ P_3 = S_3 T_3 \\ P_4 = A_{11} B_{11} \\ P_5 = A_{12} B_{21} \\ P_6 = S_4 B_{12} \\ P_7 = A_{21} T_4 \end{cases} \quad (3.c) \quad \begin{cases} U_1 = P_3 + P_5 \\ U_2 = P_1 - U_1 \\ U_3 = U_1 - P_2 \\ C_{11} = P_4 + P_5 \\ C_{12} = U_3 - P_6 \\ C_{21} = U_2 - P_7 \\ C_{22} = P_2 + U_2 \end{cases} \quad (3)$$

The eight inputs A_{ij}, B_{ij} and the four outputs C_{ij} satisfy equation (1).

The proposed method above requires 7 multiplications and $4 + 4 + 7 = 15$ linear combinations; exactly the same as the Winograd’s variant, this was proved the best possible [16].

The sequence was found with a computer-aided search within all possible linear combinations in $\mathbb{F}_2$, with one condition: the preparation phases (2.s) and (2.t) should be the same. Only 6 good combinations were found (Strassen’s is one of them). The one chosen here is the best one for squaring. There is only one equivalent sequence, which can be obtained from the above by swapping $X_{11} \leftrightarrow X_{22}$ and $X_{12} \leftrightarrow X_{21}$ for all the three matrices A, B , and C .

Then all the liftings of the sequence in $\mathbb{F}_2$ up to $\mathbb{Z}$, lifting 1 to +1 or -1, have been tested, again with the condition (2.s) $\equiv$ (2.t). The result is the sequence above, valid for any ring.

2.3 Operations for Squaring

At first we remember that matrix squaring has the same asymptotic behaviour as matrix multiplication. One side of the equivalence is obvious, because we can compute $A^2 = A \cdot A$ with a multiplication, the other side is easy too, by observing that

$$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}^2 = \begin{pmatrix} AB & 0 \\ 0 & BA \end{pmatrix}.$$

Nevertheless we can hope to save at least some operations when dealing with only one operand.

When we use formulas (2) and (3) for squaring, so that we have $A = B$, we can observe that $\forall i, S_i = T_i$. Moreover the first four products $P_1 \dots P_4$ are themselves squares, while the last three products share the three operands: A_{12}, A_{21} and S_4 . We can use only (2.s), then substitute (3.p) with:

$$\begin{cases} P_1 = S_1^2 \\ P_2 = S_2^2 \\ P_3 = S_3^2 \\ P_4 = A_{11}^2 \\ P_5 = A_{12} A_{21} \\ P_6 = S_4 A_{12} \\ P_7 = A_{21} S_4 \end{cases} \quad (4)$$

The squaring operation, then, requires half the pre-combination, since it operates on one matrix only. This was true for the original Strassen method too, but the new sequence is shorter.

LEMMA 1. *Any division free bilinear algorithm (combination; product; combination) strategy to compute the square of a 2×2 matrix, requires at least three products that are not squares.*

PROOF. Let $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ be a matrix. Consider the $\mathbb{F}_2$ -vector space of linear combinations of the 16 possible (non-commutative) products of the four entries: $\{aa, ab, ba, \dots\}$.

Let $S = \langle aa, aa + ab + ba + bb, \dots \rangle$ be the sub-space generated by all possible squares of $\mathbb{F}_2$ -linear combinations of the entries, and $Q = \langle aa + bc, ab + bd, ca + dc, dd + cb \rangle$ be the one generated by the entries of M^2 .

It is easy to verify that the intersection has dimension one $S \cap Q = \langle aa + bc + cb + dd \rangle$. Then a basis of Q must contain at least three elements that are not obtained by squaring or linear combinations of squares. $\square$

Shared Triple Product

The additive complexity (the number of linear operations) for computing the three products $A_{12}A_{21}$, $A_{21}S_4$, S_4A_{12} , is the same as the additive complexity of squaring, so that the saving extend to any recursion level thanks to the following lemmas.

LEMMA 2. *The additive complexity of computing the three products AB , BC , CA with the proposed sequence can be made the same as the additive complexity of computing three different squares (e.g. A^2, B^2, C^2).*

PROOF. By induction.

For a single recursion level: given the symmetry of (2.s) and (2.t), the linear combinations computed on the matrix A for the product AB can be used, without any re-computation for the product CA ; the same holds for the other matrices. Thus we need a set of combinations for each matrix, as if we had to compute three squares.

For any additional recursion level: let $A_{S1}, A_{S2}, \dots, C_{S4}$ be the combined sub-matrices. We need to compute the following products:

$$\begin{array}{lll} A_{S1}B_{S1} & , & B_{S1}C_{S1} & , & C_{S1}A_{S1} & , \\ A_{S2}B_{S2} & , & B_{S2}C_{S2} & , & C_{S2}A_{S2} & , \\ A_{S3}B_{S3} & , & B_{S3}C_{S3} & , & C_{S3}A_{S3} & , \\ A_{11}B_{11} & , & B_{11}C_{11} & , & C_{11}A_{11} & , \\ A_{12}B_{21} & , & B_{21}C_{S4} & , & C_{S4}A_{12} & , \\ A_{S4}B_{12} & , & B_{12}C_{21} & , & C_{21}A_{S4} & , \\ A_{21}B_{S4} & , & B_{S4}C_{12} & , & C_{12}A_{21} & . \end{array}$$

Each one of the lines above represent a triple product, thus, by induction, is equivalent to the triplet of squares or triple products used by the squaring sequence. $\square$

COROLLARY 1. *Computing the square of a matrix requires half the pre-combinations than a general product, and this is true for any recursion level.*

PROOF. The statement is obvious for one recursion level, because of symmetry. Lemma 2 extends the saving to any recursion level. $\square$

3. OPERATIONS COLLAPSING

When computing chain products or a power, we can further reduce the number of linear combinations, collating the post-combination sequence of partial results with the pre-combination needed for the next multiplication.

Let us take the simplest example: compute the product of three matrices ABC . We can compute $\tilde{A} = AB$, then $\tilde{A}C$, but we do not really need to explicitly compute all the elements of $\tilde{A}$, which is only a partial result, we can modify our sequence to obtain exactly, and only, the values needed for the next product.

The sequence below collapses formulas (3.c) and (2.s) skipping the unneeded value $\tilde{A}_{22}$, as a result we save 2 operations.

$$\begin{pmatrix} \tilde{A}_{11} \\ \tilde{A}_{21} \\ \tilde{A}_{12} \\ \tilde{S}_1 \\ \tilde{S}_2 \\ \tilde{S}_3 \\ \tilde{S}_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & -1 \\ 0 & -1 & 1 & 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & -1 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \end{pmatrix} \quad (5)$$

$$\left\{ \begin{array}{l} \tilde{A}_{11} = P_4 + P_5 \\ \tilde{A}_{12} = P_3 - P_2 - P_6 + P_5 \\ \tilde{S}_2 = P_2 + P_7 \\ \tilde{S}_1 = P_1 - P_6 \end{array} \right\} \quad (6.c)$$

$$\left\{ \begin{array}{l} \tilde{S}_3 = \tilde{S}_2 + \tilde{A}_{12} \\ \tilde{A}_{21} = \tilde{S}_1 - \tilde{S}_3 \\ \tilde{S}_4 = \tilde{S}_3 - \tilde{A}_{11} \end{array} \right\} \quad (6.s)$$

The above consideration can be generalised to any matrix chain product $\prod_{i=1}^n A_i$, saving $(n-2) \cdot 2$ combinations, but any such product should be re-implemented from scratch. So we need a more general approach.

3.1 Intermediate Representation

Equation (6) splits in two sub-sequences: (6.c) and (6.s). The first one computes four values from the products P_i , while the last three values only depend on already computed ones. This allows us to only compute the first four values, then store the intermediate result as: $\begin{pmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{S}_2 & \tilde{S}_1 \end{pmatrix}$.

This intermediate representation is tightly linked with the standard one, and we can switch from one another simply applying the invertible linear function: ψ_1 . This function only requires one addition and one subtraction, both in-place; the same occurs for its inverse that requires two subtractions.

$$\psi_1 \left(\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \right) = \begin{pmatrix} A_{11} & A_{12} \\ A_{22} - A_{21} & A_{22} + A_{12} \end{pmatrix}$$

$$\psi_1^{-1} \left(\begin{pmatrix} A_{11} & A_{12} \\ S_2 & S_1 \end{pmatrix} \right) = \begin{pmatrix} A_{11} & A_{12} \\ (S_1 - A_{12}) - S_2 & (S_1 - A_{12}) \end{pmatrix}$$

Since ψ_1 is linear, it commutes with linear combinations:

$$\forall A, B \in M_{d \times d}; \forall \alpha, \beta : \psi(\alpha A \pm \beta B) = \alpha \psi(A) \pm \beta \psi(B) \quad (7)$$

When Strassen's algorithm is used with more levels of recursion, we can also recursively define deeper transform ψ_n :

$$\psi_{n+1} \left(\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \right) = \begin{pmatrix} \psi_n(A_{11}) & \psi_n(A_{12}) \\ \psi_n(A_{21}) & \psi_n(A_{22}) \end{pmatrix}$$

Those functions work on blocks and commute with one another:

$$\forall A, B \in M_{d \times d}; \forall a, b \leq \log_2 d : \psi_a \circ \psi_b = \psi_b \circ \psi_a$$

With an abuse of notation we can say that any composition $\psi = \bigcirc_{i=1}^n \psi_i$ is linear, and equation (7) is always valid.

Carefully using one or both pre-combinations (2.s) and (6.s), the products (3.p), then one of the post-combinations (3.c) or (6.c), it is possible to build procedures taking as input the couple $\psi_a(A)$, $\psi_b(B)$ and giving the output $\psi_c(AB)$, for any needed a, b, c . All combinations are possible and

handling all of them requires much care for details, a single example will be given in subsection 3.3.

Each transform ψ_n requires additions/subtractions on half the elements of the matrix, and saves one fourth of linear operation each time the operand is used in a product. So it is worth transforming each operand used more than twice for a sequence of operations.

For example, to compute $A^7 = (A^2 \cdot A)^2 \cdot A$, we should start with $\psi(A)$.

Every intermediate result should be stored ψ -transformed, because this saves at least two operations. Conversely, the final result of the computation should be obtained with the original post-sequence (3.c), which is shorter than (6.c) followed by ψ^{-1} .

3.2 Intermediate Representation Optimality

On the side of memory footprint, the intermediate representation uses the same memory used by the usual dense unstructured matrix representation. If there are no relations *a-priori*, there is no way to squeeze the information in a smaller space.

We can then claim the following

LEMMA 3. *The additive complexity obtained with Intermediate Representation is optimal for 2×2 dense matrices stored as four entries.*

We will use two results. The first, due to Probert [16], says that the seven multiplicand must be all different combinations of values from the matrix. The second result, by Kaminski et al. [11], give us a relation between the minimal additive complexity $\delta(s)$ of the pre-combination phase for each one of the two operands and the additive complexity $\delta(c)$ of the post-combination:

$$\delta(c) \geq \delta(s) + 3. \quad (8)$$

Note that for both Strassen's original formulas ($\delta(s_S) = 5$, $\delta(c_S) = \delta(s_S) + 3$ for a grand total of $2\delta(s_S) + \delta(c_S) = 3\delta(s_S) + 3 = 18$ combinations) and Winograd's variant (i.e. $\delta(s_W) = 4$ and the total is $3\delta(s_W) + 3 = 15$) we have equality in relation (8).

PROOF. When both operands and the result use the intermediate representation, a product consists in the pre-computation (6.s), followed by the products (3.p) and recombined with (6.c).

Since we store 4 values, and we need 7 different ones, at least 3 linear combinations are required. The count of $\delta(s) = 3$ operations for equation (6.s) is then minimal.

The number of combinations in (6.c) gives $\delta(c) = 6$. We have equality in (8), so that also this value is minimal. The grand-total $2\delta(s) + \delta(c) = 3\delta(s) + 3 = 12$ is then the minimum number of linear combinations required for a product of 2×2 matrices obtained with 7 products. $\square$

We did not prove that there can not exist any other sequence, solving an equation similar to (5) for some Strassen-like product, with a smaller number of operations. We can only claim that such a sequence can not give also an optimal representation with respect to stored elements.

3.3 An Example for Chain Products

A possible application for the intermediate representation is the computation of chain products. While computing

$\prod_{i=1}^n B_i$ we may represent all partial product $R_j = \prod_{i=1}^j B_i$ ψ -transformed.

Then we will loop on the primitive $R_{j+1} \leftarrow R_j B_{j+1}$, where the two R are transformed, but B is not. We give here all details starting from the 2×2 matrices

$$\psi(R_j) = \begin{pmatrix} R_{11} & R_{12} \\ S_2 & S_1 \end{pmatrix}, \quad B_{j+1} = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}.$$

We use (6.s) on R_j and (2.t) on B ,

$$\begin{cases} S_3 = S_2 + R_{12} \\ R_{21} = S_1 - S_3 \\ S_4 = S_3 - R_{11} \end{cases} \quad \begin{cases} T_1 = B_{22} + B_{12} \\ T_2 = B_{22} - B_{21} \\ T_3 = T_2 + B_{12} \\ T_4 = T_3 - B_{11} \end{cases}$$

then the seven products in (3.p), and finally (6.c)

$$\begin{cases} P_1 = S_1 T_1 \\ P_2 = S_2 T_2 \\ P_3 = S_3 T_3 \\ P_4 = R_{11} B_{11} \\ P_5 = R_{12} B_{21} \\ P_6 = S_4 B_{12} \\ P_7 = R_{21} T_4 \end{cases} \quad \begin{cases} \tilde{R}_{11} = P_4 + P_5 \\ \tilde{R}_{12} = P_3 - P_2 - P_6 + P_5 \\ \tilde{S}_2 = P_2 + P_7 \\ \tilde{S}_1 = P_1 - P_6 \end{cases}$$

to obtain

$$\psi(R_{j+1}) = \begin{pmatrix} \tilde{R}_{11} & \tilde{R}_{12} \\ \tilde{S}_2 & \tilde{S}_1 \end{pmatrix}.$$

We saved one pre-computation and one post-computation for each product.

For the final result, when we actually need the true value of R_n we can directly use the post-computation from (3.c) in the last step, or we can apply ψ^{-1} after it.

3.4 Impact on Complexity

A referee suggested to compute the total number of operations needed for the product of two $d \times d$ matrices with $d = 2^k$ a power of 2, using k recursions. It was done by Strassen in his original paper [19], and his formula can easily generalised.

Let l be the number of linear operations needed in any Strassen-like sequence, the operation count is:

$$d^{\lg 7} \text{ prods} + \frac{l}{3} d^{\lg 7} - \frac{l}{3} d^2 \text{ adds} = \left(1 + \frac{l}{3}\right) d^{\lg 7} - \frac{l}{3} d^2.$$

Where $\lg$ is $\log_2$, the logarithm to the base 2.

Then we can resume all complexities in a single table:

Method	l	operation count
Strassen	18	$7d^{\lg 7} - 6d^2$
Winograd	15	$6d^{\lg 7} - 6d^2$
ψ -representation	12	$5d^{\lg 7} - 4d^2$
squaring	11	$\frac{14}{3}d^{\lg 7} - \frac{11}{3}d^2$
ψ -squaring	9	$4d^{\lg 7} - 3d^2$

Where ψ -operations consider both operands and result ψ -transformed.

Unfortunately the evaluation in real world implementation is much more complex. On one side the results above are unfair because in most situations addition and product (and squaring) costs are not the same. On the other side we know that frequently the recursion does not reach 2×2 matrices, but stops when some threshold is crossed.

4. COMPUTING THE POWER

There are mainly two ways to compute the power A^n of a generic matrix.

Which one is faster, depends on many parameters and implementation details, and is out of the scope of this paper. Here we will shortly outline the two strategies, and focus on the possible savings in linear operations for both algorithms by using $\psi(A)$, the intermediate representation of A .

4.1 Binary Algorithm

The classical fast exponentiation, widely known and used, is based on the binary expansion of the exponent, followed by a clever sequence of the two operations: $M \leftarrow M^2$; $M \leftarrow AM^2$.

Strassen's original algorithm would require 18 linear combination for every product or squaring.

If we use the intermediate representation for every partial result, each squaring would cost only 9 combinations. If we have $\psi(A)$ computed in a first step, every product would require $2 \times (6.s) + (6.c) = 12$ combinations.

The best results can be reached if we store all the results of the first computation from the sequence (2.s) and we keep them till the end of exponentiation. In this case also the products will require only 9 combinations, so that we halved the additive complexity with respect to Strassen's strategy.

But remember, we did not change the number of multiplications.

4.2 Polynomial Shortcut

Another way to compute the power of $A \in M_{d \times d}$ is:

- compute P , the minimal polynomial of A ;
- compute the polynomial $p_n \equiv x^n \pmod{P}$;
- evaluate the polynomial p_n on the matrix.

The polynomial p_n has degree at most $d - 1$. There are several methods to evaluate $p_n(A)$, but all of them need $d - 2$ matrix products or squarings and some linear operations.

Thanks to the linearity of the intermediate representation, we can use it for partial results in any evaluation algorithm, and we expect to save $3(d - 2)$ linear operations with the ψ -representation.

5. IMPLEMENTATIONS

The sequence proposed in §2.2 was implemented by the author for M4RI [1]: a library for linear algebra over $\mathbb{F}_2$, giving a very small (1%) but unexpected speed up for plain products.

Another unexpected result is a speed up for multiplication of rectangular matrices with the extended algebra described by D'Alberto and Nicolau [6]. Here the advantage of the new sequence with respect to Winograd's comes from the fact that the new one uses many times the sub-matrix A_{22} and only a few times A_{11} (the same for B). The former being smaller than the latter when the matrix is unevenly split. Thus the new sequence also have the same requisite as the one independently proposed by Loos [12].

5.1 Sketches on Scheduling

The new proposed sequence can be obtained from Winograd's formulas by applying permutations and sign changes, so that the scheduling work done for that sequence [3, 7, 10] can be recycled; the result is that it should be possible to substitute the new sequence into any code already im-

plementing Winograd's, basically with no need to rearrange memory usage.

All the papers above consider only two possible operations: $C \leftarrow A \cdot B$ and $C \leftarrow C + A \cdot B$. In fast GCD [14] or Jacobi symbol computation [4], another operation is typically used: $A \leftarrow A \cdot B$, where one of the two operands gets overwritten.

Only one implementation was found by the author, in the GMP-4.3 library, using Winograd's and six temporary variables. The new version of the GMP library [8] contains an implementation by the author of the new sequence. It uses only four temporaries thanks to the following schedule.

1: $U_1 \leftarrow A_{12} \cdot B_{21}$	9: $T \leftarrow T + B_{12}$	17: $U_1 \leftarrow S \cdot B_{12}$
2: $A_{22} \leftarrow A_{22} - A_{21}$	10: $A_{22} \leftarrow A_{12} \cdot T$	18: $T \leftarrow B_{22} + B_{12}$
3: $A_{12} \leftarrow A_{22} - A_{12}$	11: $A_{22} \leftarrow A_{22} - U_1$	19: $U_2 \leftarrow A_{12} \cdot T$
4: $S \leftarrow A_{12} + A_{11}$	12: $T \leftarrow T - B_{11}$	20: $A_{12} \leftarrow U_1 - A_{22}$
5: $U_2 \leftarrow A_{11} \cdot B_{11}$	13: $U_1 \leftarrow A_{21} \cdot T$	21: $A_{21} \leftarrow U_2 - A_{21}$
6: $A_{11} \leftarrow U_2 + U_1$	14: $A_{12} \leftarrow A_{12} + A_{21}$	22: $A_{22} \leftarrow U_2 - A_{22}$
7: $T \leftarrow B_{22} - B_{21}$	15: $A_{21} \leftarrow U_1 + A_{22}$	
8: $U_2 \leftarrow A_{22} \cdot T$	16: $A_{22} \leftarrow A_{22} - U_2$	

Table 1: Scheduling for the $A \leftarrow A \cdot B$ operation.

If B can be overwritten as well, the temporary T can be removed, using B_{21} for it. Anyway, the main purpose of this section is not to exhaust the subject of scheduling, but the opposite: to remark the fact that a lot of work can be done on new sequences, probably exploring all of them as collected in Appendix R.

For example the sequence in table 1 is not exactly the one described for squaring, because some sign was changed. . .

6. CONCLUSIONS

We have shown, in §2.2, a new sequence for Strassen-like matrix multiplication. This new sequence is optimal with respect to multiplicative and additive complexity for generic 2×2 matrices and for recursive use.

Thanks to the additional property of symmetry, half of the preliminary operations can be saved when the product involves one operand only. The squaring case also has the maximal number of recursive multiplications being themselves squares. Again the new sequence is optimal.

The sequence can be shortened even more with some linear pre-computations. We have shown an in-place transform for matrices, requiring 2 operations per recursion level, after which any product costs 3 operations less. Transformed and standard matrices can be mixed, and some gain can be achieved for chain products even if none of the operands is transformed in advance.

We then propose the use of our new sequence for every implementation of Strassen's matrix multiplication. It is not worse than the widely used Winograd sequence for simple multiplications, but it can give performance gain for squaring, chain products and general polynomial computation.

Acknowledgements

The authors want to thank the anonymous reviewers for their valuable suggestions and corrections.

7. REFERENCES

- [1] Martin Albrecht and Gregory Bard. *The M4RI Library - Version 20090409*. The M4RI Team, 2009.

Dimension	2	3	4	5	6	7	8	9	16
Product [13, p.30]	7	23	46	93	141	235	316	473	2212
Naïve squaring (9)	5	18	46	95	171	280	428	621	3736
Combined squaring	5	18	41	93	141	235	302	473	2156

Table 2: Number of ring products required for squaring small matrices.

- [2] Dario Bini and Victor Pan. *Polynomial and Matrix Computations*, volume 1. Birkhauser, Boston, USA, 1994.
- [3] Brice Boyer, Jean-Guillaume Dumas, Clément Pernet, and Wei Zhou. Memory efficient scheduling of Strassen-Winograd’s matrix multiplication algorithm. In *ISSAC ’09: Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 55–62, Seoul, Korea, 2009. ACM.
- [4] Richard P. Brent and Paul Zimmermann. An $O(M(n) \log n)$ algorithm for the Jacobi symbol. In Guillaume Hanrot, François Morain, and Emmanuel Thomé, editors, *Proceedings of the 9th Algorithmic Number Theory Symposium (ANTS-IX)*, volume 6197 of *LNCS*, Nancy, France, July 19-23, 2010. Springer.
- [5] Nader H. Bshouty. On the additive complexity of 2×2 matrix multiplication. *Information processing letters*, 56(6):329–336, December 1995.
- [6] Paolo D’Alberto and Alexandru Nicolau. Adaptive Winograd’s matrix multiplications. *Transactions on Mathematical Software*, 36(1):1–23, 2009.
- [7] Craig C. Douglas, Michael Heroux, Gordon Sliselman, and Roger M. Smith. GEMMW: A portable level 3 BLAS Winograd variant of Strassen’s matrix–matrix multiply algorithm. *Journal of Computational Physics*, 110(1):1–10, 1994.
- [8] Torbjörn Granlund. *GNU MP – The GNU Multiple Precision Arithmetic Library*. The GMP development team, 2010.
- [9] Te C. Hu. Revised matrix algorithms for shortest paths. *SIAM Journal on Applied Mathematics*, 15(1):207–218, Jan 1967.
- [10] Steven Huss-Lederman, Elaine M. Jacobson, Jeremy R. Johnson, Anna Tsao, and Thomas Turnbull. Strassen’s algorithm for matrix multiplication: Modeling, analysis, and implementation. Technical Report CCS-TR-96-17, Center for Computing Sciences, November 15 1996.
- [11] Michael Kaminski, David G. Kirkpatrick, and Nader H. Bshouty. Addition requirements for matrix and transpose matrix product. *Journal of Algorithms*, 9:354–364, 1988.
- [12] Sarah M. Loos and David S. Wise. Strassen’s matrix multiplication relabeled. <http://src.acm.org/loos/loos.html>, December 2009.
- [13] Marc Mezzarobba. Génération automatique de procédures numériques pour les fonctions D-finies. Master’s thesis, Master parisien de recherche en informatique, August 2007.
- [14] Niels Möller. On Schönhage’s algorithm and subquadratic integer GCD computation. *Mathematics of Computation*, 77(261):589–607, 2008.
- [15] Vittorio Ottaviani, Alberto Zanon, and Massimo Regoli. Conjugation as public key agreement protocol in mobile cryptography. In *SECRYPT*, Athens, Greece, 2010.
- [16] Robert L. Probert. On the additive complexity of matrix multiplication. *SIAM Journal on Computing*, 5(2):187–203, June 1976.
- [17] Eligijus Sakalauskas, Povilas Tvarijonas, and Andrius Raulynaitis. Key agreement protocol (KAP) using conjugacy and discrete logarithm problems in group representation level. *Informatica*, 18(1):115–124, 2007.
- [18] René Schott and George Stacey Staples. Nilpotent adjacency matrices, random graphs and quantum random variables. *Journal of Physics A: Mathematical and Theoretical*, 41(15), 2008.
- [19] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.
- [20] Abraham Waksman. On Winograd’s algorithm for inner products. *IEEE Transactions on Computers*, 19(4):360–361, April 1970.
- [21] Shmuel Winograd. On multiplication of 2×2 matrices. *Linear Algebra and Application*, 4:381–388, 1971.

APPENDIX

Q. COMMUTATIVE MATRIX SQUARING

For very small matrices, recursive methods are often too expensive, particularly when we can use commutativity of the base ring.

For 2×2 and 3×3 matrices, we trivially have

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^2 = \begin{pmatrix} a^2 + bc & b(a+d) \\ c(a+d) & d^2 + bc \end{pmatrix}$$

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}^2 = \begin{pmatrix} a^2 + bd + cg & b(a+e) + ch & c(a+i) + bf \\ d(a+e) + fg & e^2 + bd + fh & cd + f(e+i) \\ g(a+i) + dh & h(e+i) + bg & i^2 + fh + cg \end{pmatrix}$$

The formula can be generalised to any dimension requiring

$$d \text{ squares, } d^3 - d^2 - \binom{d}{2} \text{ products, } d^3 - d^2 - \binom{d}{2} \text{ additions.} \quad (9)$$

This trivial optimisation should be compared to Waksman multiplication [20] for $d \times d$ matrices, and to recursive use of Strassen-like algorithm. It wins only for dimension 2 and 3. Nevertheless its combined use with one or more recursions of the sequence proposed in §2.2 can give the best algorithm for squaring very small matrices if we count the total number of ring products required. In the table 2 we compare our matrix squaring with the best known algorithms to compute the product of two matrices.

In particular we should notice that:

- for the 5×5 squaring, Waksman requires 93 products, equation (9) proposes 90 products plus 5 squares, which may be better in some cases;

- for the 6×6 case, Waksman needs 141 multiplications (sums $\cong 480$), our new sequence splits in four 3×3 squares (using (9)), and three 3×3 products (Waksman), totalling $23 \cdot 3 + 15 \cdot 4 = 129$ products and $3 \cdot 4 = 12$ squarings; same number of operations (fewer sums $\cong 390$), but with a better product/square ratio.

R. OTHER RESULTS IN GF(2)

There are some obvious transformations which allow us to take a Strassen-like matrix multiplication algorithm and to obtain another [5].

Let J be the matrix $J = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$: the two transformations $A \cdot B = (B^T \cdot A^T)^T$ and $A \cdot B = J((JAJ) \cdot (JBJ))J$ (which is the symmetry we made mention of in §2.2), are the only two preserving both the kind of operations and the existing symmetry.

By adding other transformations not requiring additional linear operations, namely $A \cdot B = J((JA) \cdot B) = (A \cdot (BJ))J$, we can group all the possible Strassen-like sequences in four equivalence classes:

1. one containing the proposed sequence, the symmetric equivalent, Winograd's and 5 different ones;
2. one containing the original Strassen's and 3 more;
3. one with $A_{11} \cdot (B_{11}+B_{12})$; $(A_{12}+A_{22}) \cdot (B_{12}+B_{22})$; $(A_{12}+A_{21}+A_{22}) \cdot (B_{12}+B_{21})$; $A_{12} \cdot (B_{21}+B_{22})$; $(A_{21}+A_{22}) \cdot B_{21}$; $A_{21} \cdot (B_{11}+B_{21})$; $(A_{11}+A_{12}+A_{21}+A_{22}) \cdot B_{12}$, and 15 more sequences
4. and the one represented by $(A_{11}+A_{12}+A_{22}) \cdot (B_{11}+B_{12}+B_{22})$; $A_{11} \cdot (B_{12}+B_{22})$; $A_{22} \cdot B_{21}$; $(A_{11}+A_{12}+A_{21}+A_{22}) \cdot (B_{11}+B_{12})$; $(A_{12}+A_{22}) \cdot (B_{11}+B_{12}+B_{21}+B_{22})$; $A_{21} \cdot B_{11}$; $(A_{11}+A_{12}) \cdot B_{22}$, containing also 7 other sequences.

Totalling only 36 possible combinations in $\mathbb{F}_2$. Obviously every multiplication sequence in characteristic zero must be a lifting of one of them.

Any search for possible schedulings should test at least one sequence in each one of the classes above, because Strassen's and Winograd's are not the only sequences available.

T. TIMINGS

The new sequence has many possible applications, and the intermediate representation adds many possible variants. It is quite difficult to implement all possible variations and to show a graph giving all needed informations at a glance.

The author implemented some simple functions to compute 2×2 matrix product and squaring, using some different strategies. The entries are integers, for this test implementation the `mpz` type provided by GMP-5.0.1 [8] was used. Timings have been measured on a 32-bits Centrino, running Debian GNU/Linux. With different architectures the actual numbers can be different, but the shapes should be similar.

Figure 1 shows relative timings for matrix-matrix multiplication, comparing four different algorithms. The naïve 8-products algorithm is used as a reference, it's timings are normalised as 100%. Two algorithms are indistinguishable: Winograd, and the new proposed sequence. This was expected, because the operation count is exactly the same; the threshold between the naïve algorithm and Strassen-like is somewhere around 500 bits.

The fourth algorithm named "psi multiplication" is the product with both operands and the result using the inter-

mediate representation, three linear operations are saved. As a result it wins on the naïve algorithm a little bit earlier, and it is a little bit faster than plain Winograd for all entries sizes.

The second graph in Figure 2 shows relative timings for squaring. The algorithm are named in the key sorted by their timings on the right side of the graph (operands with entries of 800-bits).

Naïve multiplication and naïve squaring are the same algorithm. Using a single operand, squaring compute the squares of two entries. Squaring an integer is faster than performing a multiplication, thanks to the underlying GMP library, that's why the timings are different.

Winograd uses only one squaring on the entries, and its threshold with respect to naïve squaring is around 700 bits, much larger than the multiplication threshold.

Used for squaring, the new sequence is much faster than Winograd. For this implementation and this dimension, we can argue that the speed saving mostly come from the high number of multiplications replaced by squarings on entries. The "psi-squaring" uses the intermediate representation and saves some more linear operation; we can observe that the additional speed-up is not large.

The real winner in this graph is the algorithm exploiting the commutativity of integers, the one described in Appendix Q. We remember here that it can only win for 2×2 or 3×3 matrices, where the total number of non-linear operations (products or squarings) can be reduced.

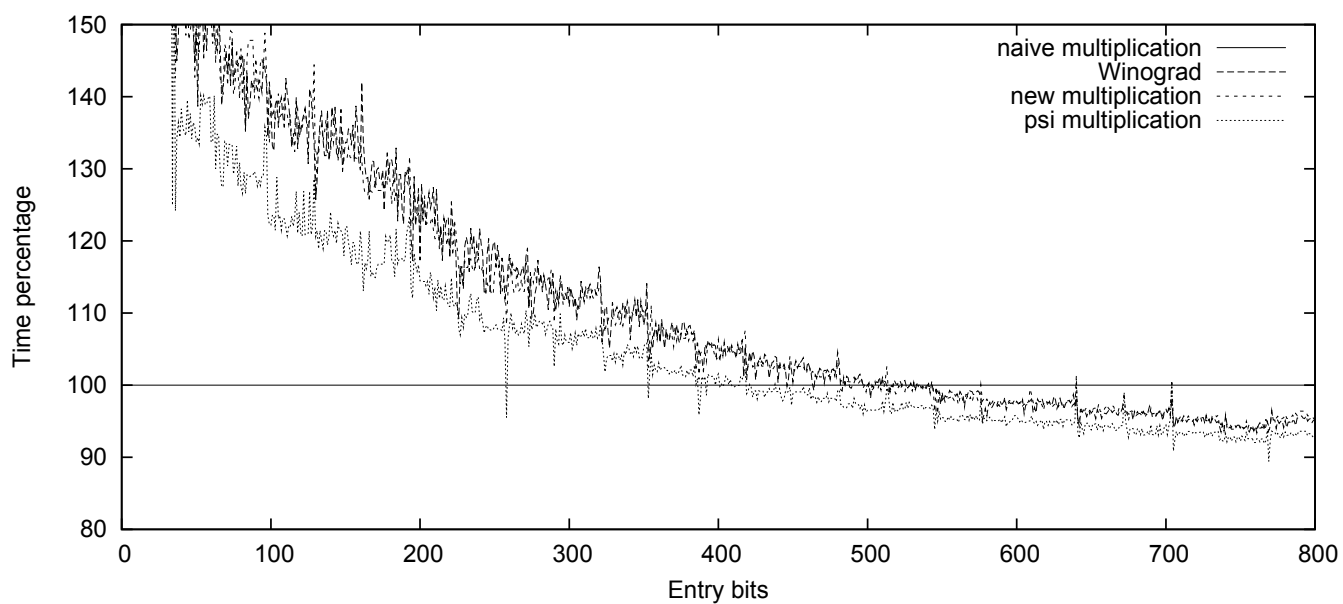


Figure 1: Matrix-matrix product timing comparisons.

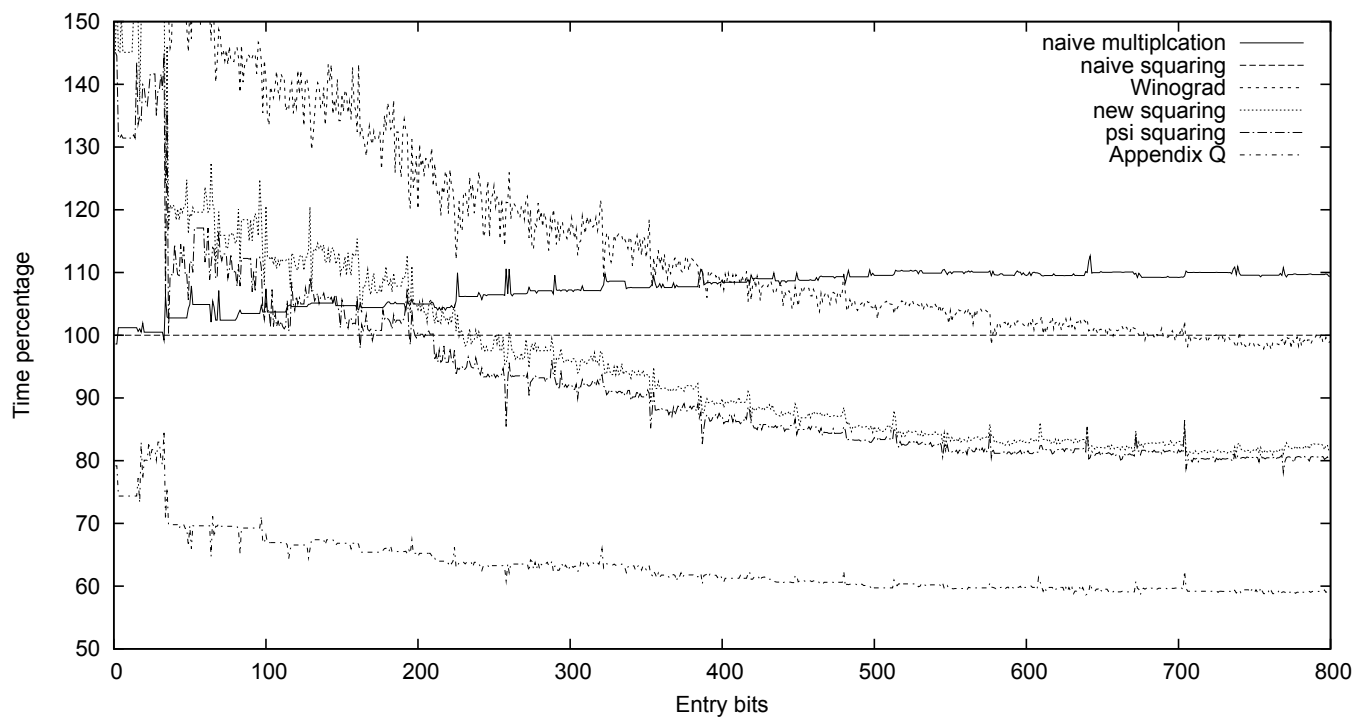


Figure 2: Matrix squaring timing comparisons.

Computing Specified Generators of Structured Matrix Inverses

Claude-Pierre Jeannerod
INRIA

Laboratoire LIP (CNRS, ENS de Lyon, INRIA,
UCBL), Université de Lyon, France
claude-pierre.jeannerod@ens-lyon.fr

Christophe Moulleron
ENS de Lyon

Laboratoire LIP (CNRS, ENS de Lyon, INRIA,
UCBL), Université de Lyon, France
christophe.moulleron@ens-lyon.org

ABSTRACT

The asymptotically fastest known divide-and-conquer methods for inverting dense structured matrices are essentially variations or extensions of the Morf/Bitmead-Anderson algorithm. Most of them must deal with the growth in length of intermediate generators, and this is done by incorporating various generator compression techniques into the algorithms. One exception is an algorithm by Cardinal, which in the particular case of Cauchy-like matrices avoids such growth by focusing on well-specified, already compressed generators of the inverse. In this paper, we extend Cardinal's method to a broader class of structured matrices including those of Vandermonde, Hankel, and Toeplitz types. Besides, some first experimental results illustrate the practical interest of the approach.

Categories and Subject Descriptors

I.1.2 [Computing Methodologies]: Symbolic and Algebraic Manipulation—*Algebraic Algorithms*

General Terms

Algorithms, Theory

Keywords

Structured linear algebra, matrix inversion

1. INTRODUCTION

Since [10], a classical way of exploiting the structure of dense matrices is via the displacement rank approach: typically, $n \times n$ matrices are represented by pairs (G, H) of $n \times \alpha$ matrices such that $\mathcal{L}(A) = GH^T$ for some linear operator $\mathcal{L}$ called a displacement. Classical choices for $\mathcal{L}$ are Stein's displacement $\Delta[M, N] : A \mapsto A - MAN$ and Sylvester's displacement $\nabla[M, N] : A \mapsto MA - AN$. With respect to a given displacement, (G, H) is called a *generator* of length α for A , and A is considered to be structured when α is “small” (in

a sense that depends on the context) compared to n . According to the unified treatment [19], many of the structures encountered in practice are covered by the following operator matrices: for a field $\mathbb{K}$ and a positive integer n , let

$$M, N \in \{\mathbb{D}(x), \mathbb{Z}_{n,\varphi}, \mathbb{Z}_{n,\psi}^T\}, \quad x \in \mathbb{K}^n, \quad \varphi, \psi \in \mathbb{K}, \quad (1)$$

with $\mathbb{D}(x)$ the diagonal matrix whose entry (i, i) is the i th coefficient x_i of vector x , and $\mathbb{Z}_{n,\varphi}$ the $n \times n$ unit φ -circulant matrix having a φ in position $(1, n)$, ones in positions $(i + 1, i)$, and zeros everywhere else.

When a structured matrix $A \in \mathbb{K}^{n \times n}$ is invertible, its inverse A^{-1} is known to be structured too, and some asymptotically fast algorithms are available for computing length- α generators for A^{-1} and linear system solutions, whose costs in terms of operations in $\mathbb{K}$ are in $O(\alpha^2 n)$ (see [19] and the references therein) and, since more recently, in $O(\alpha^{\omega-1} n)$ (see [2, 3]). (Here and hereafter the O notation hides logarithmic factors, and ω denotes a feasible exponent for matrix multiplication over $\mathbb{K}$.) Such algorithms are essentially variations or extensions of the Morf/Bitmead-Anderson (MBA) divide-and-conquer approach [13, 1]. In practice, they apply to important types of structures like those of (1). However, most of these algorithms must deal with the growth in length of intermediate generators, and this is done by recursively using a generator compression stage which, given matrices $G, H \in \mathbb{K}^{n \times \beta}$ such that GH^T has rank $\alpha \leq \beta$, computes matrices G_c, H_c that satisfy $G_c H_c^T = GH^T$ but now have exactly α columns; see [16, 15, 17, 11, 12] and [19, §4.6].

One exception is a variant of MBA due to Cardinal [4, 5]: assuming Sylvester's displacement equation

$$\nabla[M, N](A) = GH^T \quad (2)$$

and in the particular case where both M and N are diagonal (Cauchy-like structure), Cardinal's algorithm completely avoids generator compression by directly computing

$$Y = -A^{-1}G, \quad Z = A^{-T}H. \quad (3)$$

As already noted in [9] (and this is readily verified by pre- and postmultiplying (2) with the inverse of A), the matrix pair (Y, Z) is a $\nabla[N, M]$ -generator of length α for A^{-1} . Due to its very special form, we shall call it a *specified generator* for the inverse of A .

The goal of this paper is to extend Cardinal's algorithm beyond the Cauchy-like structure and to show that, in MBA and for Sylvester's displacement, generator compression can be systematically avoided by targeting a specified generator for the inverse, rather than just an arbitrary one of length α . More precisely, our three main contributions can be summarized as follows:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

First, we propose a recursive formula that allows to factor a specified generator of the inverse for A in terms of specified generators for the inverse of its upper-left block A_{11} and for the inverse of the Schur complement of A_{11} in A .

Second, we show how to reduce the computation of specified inverse generators for the structures defined in (1) to the computation of specified inverse generators for the three basic cases below:

$$(M, N) \in \left\{ (\mathbb{D}(x), \mathbb{D}(y)), (\mathbb{D}(x), \mathbb{Z}_{n,0}^T), (\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^T) \right\}. \quad (4)$$

For each of those three structures, which are of the Cauchy-like, Vandermonde-like, and Hankel-like types, respectively, we further give and analyze explicit algorithms for computing a specified generator of the inverse. These algorithms are compression-free and thus, in that sense, simpler to analyze and implement than traditional MBA variants. Moreover, although removing generator compression does not affect the overall asymptotic costs, it yields smaller dominant terms.

Third, we report on a first set of experiments done with our C++ implementation of MBA and of several of the new compression-free algorithms. For the Cauchy-like structure, for example, the speed-ups compared to MBA are by a factor from 4.6 to 6.7. This suggests that our extension of Cardinal's compression-free approach may yield algorithms that are not only simpler but also significantly faster in practice.

Outline of the paper. After some notation and preliminaries in §2, some properties of specified generators are studied in §3. Then §4 gives a compression-free algorithm for the case where M and N^T are lower triangular. The algorithm is specialized to the Cauchy-like and Vandermonde-like structures in §4.1 and §4.2, and then extended in §4.3 to the irregular Hankel-like case $(M, N) = (\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^T)$. Experiments are reported in §5 and we conclude in §6.

2. NOTATION AND PRELIMINARIES

Here and hereafter, $\mathbb{I}_n$ is the identity matrix of order n , $\mathbf{e}_{n,i}$ is the i th unit vector of $\mathbb{K}^n$, and $\mathbb{J}_n$ is the reflexion matrix of order n , whose (i, j) entry is 1 if $i + j = n + 1$, and 0 otherwise. For $A \in \mathbb{K}^{n \times m}$, a_{ij} denotes its (i, j) entry and $\mathbf{a}_j$ its j th column. Also, for $\alpha \leq m$, we write $A^{\rightarrow \alpha}$ for the matrix $[\mathbf{a}_1, \dots, \mathbf{a}_\alpha] \in \mathbb{K}^{n \times \alpha}$.

Given positive integers n_1 and n_2 such that $n_1 + n_2 = n$, we partition A, G, H, M, N into $n_i \times n_j$ or $n_i \times \alpha$ blocks as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}, \quad H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}, \quad (5)$$

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}, \quad N = \begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix}.$$

We shall write μ and ν for the rank of, respectively, M_{12} and N_{21} . Consequently, those two matrices can be written

$$M_{12} = U_1 V_2^T, \quad N_{21} = U_2 V_1^T \quad (6)$$

for some full column rank matrices $U_1 \in \mathbb{K}^{n_1 \times \mu}$, $V_2 \in \mathbb{K}^{n_2 \times \mu}$, $U_2 \in \mathbb{K}^{n_2 \times \nu}$, and $V_1 \in \mathbb{K}^{n_1 \times \nu}$.

The Schur complement of A_{11} in A is written S :

$$S = A_{22} - A_{21} A_{11}^{-1} A_{12}.$$

Recall that S is nonsingular if A_{11} and A are nonsingular; if A is strongly regular then so are A_{11} and S . Finally, let

$$E = \begin{bmatrix} \mathbb{I}_{n_1} & \\ -A_{21} A_{11}^{-1} & \mathbb{I}_{n_2} \end{bmatrix}, \quad F = \begin{bmatrix} \mathbb{I}_{n_1} & -A_{11}^{-1} A_{12} \\ & \mathbb{I}_{n_2} \end{bmatrix}. \quad (7)$$

From $EAF = \text{diag}(A_{11}, S)$, we deduce the following classical recursive factorization of the inverse of A [19, p. 157]:

$$A^{-1} = F \begin{bmatrix} A_{11}^{-1} & \\ & S^{-1} \end{bmatrix} E. \quad (8)$$

2.1 Properties of Sylvester's displacement

The properties below show how to deduce from a $\nabla[M, N]$ -generator for A a generator for various matrices related to A . All of them appear in/follow immediately from [18, 19].

Generation of the transpose. Let A, G, H be as in (2). By transposing the identity $MA - AN = GH^T$, we obtain

$$\nabla[N^T, M^T](A^T) = -HG^T, \quad (9)$$

so that the pair $(-H, G)$ is a $\nabla[N^T, M^T]$ -generator for A^T .

Generation of products. One has the following classical rule for generating matrix products [19, p. 10]:

$$\nabla[M, N](AB) = \nabla[M, P](A)B + A\nabla[P, N](B), \quad (10)$$

for any matrix P of conforming dimensions. Applying this rule twice, we can straightforwardly deduce explicit formulas for generating products of three matrices:

LEMMA 1. Let A, G, H be as in (2) and, for two matrices P_1 and P_2 , let

$$\tilde{A} = P_1 A P_2. \quad (11)$$

If $\nabla[\tilde{M}, \tilde{M}](P_1) = G_{P_1} H_{P_1}^T$ and $\nabla[N, \tilde{N}](P_2) = G_{P_2} H_{P_2}^T$ then

$$\nabla[\tilde{M}, \tilde{N}](\tilde{A}) = \tilde{G} \tilde{H}^T,$$

$$\tilde{G} = [P_1 G | G_{P_1} | P_1 A G_{P_2}], \quad \tilde{H} = [P_2^T H | P_2^T A^T H_{P_1} | H_{P_2}]. \quad (12)$$

As an example, let us mention three special cases which we will use later: assuming $M, N \in \{\mathbb{Z}_{n,\varphi}, \mathbb{Z}_{n,\varphi}^T\}$, let first

$$(P_1, P_2) = (\mathbb{I}_n, \mathbb{J}_n) \quad \text{and} \quad (\tilde{M}, \tilde{N}) = (M, N^T).$$

Then obviously $\nabla[\tilde{M}, \tilde{M}](P_1)$ is zero and, using the facts that $\mathbb{J}_n^2 = \mathbb{I}_n$ and $\mathbb{J}_n \mathbb{Z}_{n,\varphi} \mathbb{J}_n = \mathbb{Z}_{n,\varphi}^T$ (see [19, p. 24]), we deduce that $\nabla[N, \tilde{N}](P_2)$ is zero as well. Consequently, since $\mathbb{J}_n$ is symmetric, applying (12) yields

$$\nabla[M, N^T](A \mathbb{J}_n) = G(\mathbb{J}_n H)^T. \quad (13a)$$

Similarly, exchanging the roles of P_1 and P_2 yields

$$\nabla[M^T, N](\mathbb{J}_n A) = (\mathbb{J}_n G) H^T, \quad (13b)$$

while taking $P_1 = P_2 = \mathbb{J}_n$ gives

$$\nabla[M^T, N^T](\mathbb{J}_n A \mathbb{J}_n) = (\mathbb{J}_n G)(\mathbb{J}_n H)^T. \quad (13c)$$

Generation of submatrices. From (2) and (6) and the partitioning into blocks we deduce that, for $i, j \in \{1, 2\}$, submatrix A_{ij} satisfies the following matrix equation

$$\nabla[M_{ij}, N_{ij}](A_{ij}) = G_{ij} H_{ij}^T, \quad (14)$$

where, in particular (see for example [18, Proposition 4.4]),

$$G_{11} = [G_1 | -U_1 | A_{12} U_2] \in \mathbb{K}^{n_1 \times (\alpha + \mu + \nu)}, \quad (15a)$$

$$H_{11} = [H_1 | A_{21}^T V_2 | V_1] \in \mathbb{K}^{n_1 \times (\alpha + \mu + \nu)}. \quad (15b)$$

Generation of Schur complements. By combining [18, Proposition 4.5] with (6), we have the following description of the structure of the Schur complement S of A_{11} in A :

$$\nabla[M_{22}, N_{22}](S) = G_S H_S^T, \quad (16)$$

with G_5 and H_5 the two matrices in $\mathbb{K}^{n_2 \times (\alpha + \mu + \nu)}$ given by

$$G_5 = [G_2 - A_{21}A_{11}^{-1}G_1 | A_{21}A_{11}^{-1}U_1 | -SU_2], \quad (17a)$$

$$H_5 = [H_2 - A_{12}^T A_{11}^{-T} H_1 | S^T V_2 | A_{12}^T A_{11}^{-T} V_1]. \quad (17b)$$

When the operator matrices M and N^T are lower triangular, one has $\mu = \nu = 0$ and the above formulas for generating the Schur complement can thus be simplified as follows (see [8, Theorem 2.3], [14, Lemma 3.1], [19, §5.4]):

$$G_5 = G_2 - A_{21}A_{11}^{-1}G_1, \quad H_5 = H_2 - A_{12}^T A_{11}^{-T} H_1. \quad (18)$$

2.2 Computing with basic structures

We conclude our preliminaries by reviewing three basic invertible displacement operators that we shall repeatedly use in the sequel, as well as some associated cost functions. Here we assume that (2) holds in the rectangular case, that is, for $A \in \mathbb{K}^{n \times m}$, $G \in \mathbb{K}^{m \times \alpha}$, and $H \in \mathbb{K}^{m \times \alpha}$; this assumption will allow us to handle off-diagonal blocks in Section 4. Recall also that $\nabla[M, N]$ is invertible if and only if the spectra of M and N are disjoint [19, p. 123].

Cauchy-like structure. For $x \in \mathbb{K}^n$ and $y \in \mathbb{K}^m$, assume

$$M = \mathbb{D}(x), \quad N = \mathbb{D}(y), \quad x_i \neq y_j \text{ for all } (i, j). \quad (19a)$$

Then $\nabla[M, N]$ is invertible and it is known [7] (see also [19, p. 8] and [20, Lemma 2.1]) that (2) is equivalent to

$$A = \sum_{j=1}^{\alpha} \mathbb{D}(g_j) \mathbb{C}(x, y) \mathbb{D}(h_j), \quad (19b)$$

with $\mathbb{C}(x, y)$ the n by m Cauchy matrix $[1/(x_i - y_j)]_{i,j}$.

Vandermonde-like structure. For $x \in \mathbb{K}^n$, assume now

$$M = \mathbb{D}(x), \quad N = \mathbb{Z}_{m,0}^T, \quad x_i \neq 0 \text{ for all } i. \quad (20a)$$

Then $\nabla[M, N]$ is invertible and, in this case, A can be recovered as follows (see [19, Example 4.4.6(d)]):

$$A = \mathbb{D}(x)^{-1} \sum_{j=1}^{\alpha} \mathbb{D}(g_j) \mathbb{V}(x^{-1}, m) \mathbb{U}(h_j), \quad (20b)$$

where $\mathbb{V}(x^{-1}, m)$ is the n by m Vandermonde matrix whose (i, j) entry equals $(1/x_i)^{j-1}$, and $\mathbb{U}(h_j)$ is the m by m upper triangular Toeplitz matrix whose first row is h_j^T .

Hankel-like structure. Assume finally that

$$M = \mathbb{Z}_{n,1}, \quad N = \mathbb{Z}_{m,0}^T. \quad (21a)$$

Since $\mathbb{Z}_{n,1}$ and $\mathbb{Z}_{m,0}^T$ have disjoint spectra, $\nabla[M, N]$ is invertible. In addition, we can recover A as follows:

$$A = \sum_{j=1}^{\alpha} \mathbb{T}^{n \times m}(g_j) \mathbb{L}(h_j) \mathbb{J}_m, \quad (21b)$$

where, for $x \in \mathbb{K}^n$, $\mathbb{T}^{n \times m}(x)$ is the n by m Toeplitz matrix $[x_{1+(i-j+m) \bmod n}]_{i,j}$, and where $\mathbb{L}(h_j)$ is the m by m lower triangular Toeplitz matrix whose first column is h_j . A proof of (21b) is given in Appendix A of a draft of this paper.¹

Cost functions. Our algorithms in the next sections will essentially require the ability to efficiently evaluate products of the form AB and $A^T B$, where A has one of the three basic structures above, and B consists of one or several vectors.

In order to relate the costs of our algorithms in Section 4 to the costs of such products, we introduce the following functions. For the Cauchy-like structure (19a), let $\text{MM}_C : \mathbb{N}_{>0} \times \mathbb{N}_{>0} \times \mathbb{N}_{>0} \rightarrow \mathbb{R}_{\geq 0}$ be such that, for $A \in \mathbb{K}^{n \times n}$ given by the right hand-side of (19b) and $B \in \mathbb{K}^{n \times \beta}$, the products AB and $A^T B$ can be computed using at most $\text{MM}_C(\alpha, n, \beta)$

operations in $\mathbb{K}$. We define the functions MM_V and MM_H in a similar way for, respectively, the Vandermonde-like and Hankel-like structures. Also, when $\beta = \alpha$ we shall simply write $\text{MM}_*(\alpha, n)$, for $*$ = C, V, H.

Following [6, p. 242], we write $M(n)$ for the cost of multiplying two polynomials of degree less than n over $\mathbb{K}[x]$, and we assume that $M(n)$ is “superlinear,” that is, $M(n)/n$ is nondecreasing.

It is known (see for example [19]) that $\mathbb{C}(x, y)^T$ is $-\mathbb{C}(y, x)$ and that multiplying $\mathbb{C}(x, y)$, $\mathbb{V}(x, n)$, or $\mathbb{V}(x, n)^T$ by a vector can be done in time $O(M(n) \log(n))$ via (transposed) multi-point evaluation. Hence by a straightforward application of the summation formulas (19b), (20b), and (21b), one has

$$\text{MM}_*(\alpha, n, 1) \in O(\alpha M(n) \log(n)) \quad \text{for } * = C, V,$$

$$\text{MM}_H(\alpha, n, 1) \in O(\alpha M(n)).$$

We shall also use the three properties given below:

LEMMA 2. Let $k, \ell \in O(1)$. Then

$$\text{MM}_V(\alpha + k, n, \alpha) \in \text{MM}_V(\alpha, n) + O(\alpha M(n) \log(n)), \quad (22a)$$

$$\text{MM}_H(\alpha + k, n, \alpha) \in \text{MM}_H(\alpha, n) + O(\alpha M(n)), \quad (22b)$$

and, for $*$ = C, V, H,

$$\text{MM}_*(k\alpha, n, \ell\alpha) \in k\ell \text{MM}_*(\alpha, n) + O(\alpha n). \quad (23)$$

PROOF. To get (22) note that, for all $*$, $\text{MM}_*(\alpha + k, n, \alpha)$ is in $\text{MM}_*(\alpha, n, \alpha) + \text{MM}_*(k, n, \alpha) + O(\alpha n)$. Indeed, one can evaluate our sum of $\alpha + k$ products by adding the first α terms and the last k terms separately, and then combining the two intermediate results. Since moreover $\text{MM}_*(k, n, \alpha) \leq \alpha \text{MM}_*(k, n, 1)$, (22a) and (22b) follow from the complexities of $\text{MM}_V(\alpha, n)$ and $\text{MM}_H(\alpha, n)$ mentioned above. To establish (23), notice that a sum of $k\alpha$ terms for $\ell\alpha$ vectors can be evaluated via k sums of α terms for α vectors plus a final sum in $O(\alpha n)$, repeated ℓ times. $\square$

Finally, we assume as for $M(n)$ that the functions $\text{MM}_*(\cdot, n)$ are superlinear. This assumption will allow us to simplify the cost bounds of the algorithms of Section 4 and can be easily supported by “naive” implementations in $O(\alpha^2 n)$ as those used in Section 5.

3. PROPERTIES OF SPECIFIED GENERATORS OF THE MATRIX INVERSE

3.1 Recovery after matrix transformations

We recalled in Section 2 some formulas for generating the matrix $\tilde{A} \in \{A^T, P_1 A P_2\}$ from some generators of the matrix A . Conversely, we give in the theorem below some formulas for recovering specified generators of the inverse of A from specified generators of the inverse of $\tilde{A}$.

THEOREM 1. Let $A \in \mathbb{K}^{n \times n}$ be invertible and let $G, H \in \mathbb{K}^{n \times \alpha}$ and $Y, Z \in \mathbb{K}^{n \times \alpha}$ be as in (2) and (3). Let $\tilde{A} \in \mathbb{K}^{n \times n}$ be invertible and, for $\tilde{G}, \tilde{H} \in \mathbb{K}^{n \times \beta}$, $\beta \geq \alpha$, define

$$\tilde{Y} = -\tilde{A}^{-1} \tilde{G}, \quad \tilde{Z} = \tilde{A}^{-T} \tilde{H}.$$

Then

- for $\tilde{A} = A^T$ and $(\tilde{G}, \tilde{H}) = (-H, G)$, one has

$$Y = -\tilde{Z}, \quad Z = \tilde{Y};$$

¹<http://prunel.ccsd.cnrs.fr/ensl-00450272/en/>

- for $\tilde{A} = P_1 A P_2$ with $P_1, P_2 \in \mathbb{K}^{n \times n}$ invertible, and for $\tilde{G}, \tilde{H}$ as in (12), one has

$$Y = P_2 \tilde{Y}^{\rightarrow \alpha}, \quad Z = P_1^T \tilde{Z}^{\rightarrow \alpha}.$$

PROOF. In the first case, $\tilde{Y} = -(A^{-T})(-H) = A^{-T}H = Z$ and $\tilde{Z} = (A^T)^{-T}G = A^{-1}G = -Y$. Now, in the case where $\tilde{A} = P_1 A P_2$ Lemma 1 implies that the first α columns of $\tilde{Y}$ are $\tilde{Y}^{\rightarrow \alpha} = -(P_1 A P_2)^{-1} P_1 G = P_2^{-1} Y$. Similarly, the first α columns of $\tilde{Z}$ are $\tilde{Z}^{\rightarrow \alpha} = (P_1 A P_2)^{-T} P_2^T H = P_1^{-T} Z$. $\square$

For example, when $P_1, P_2 \in \{\mathbb{I}_n, \mathbb{J}_n\}$, it follows from (12) that $\beta = \alpha$. Consequently, Theorem 1 yields

$$(Y, Z) = (\mathbb{J}_n \tilde{Y}, \tilde{Z}) \quad \text{if } \tilde{A} = A \mathbb{J}_n, \quad (25a)$$

$$(Y, Z) = (\tilde{Y}, \mathbb{J}_n \tilde{Z}) \quad \text{if } \tilde{A} = \mathbb{J}_n A, \quad (25b)$$

$$(Y, Z) = (\mathbb{J}_n \tilde{Y}, \mathbb{J}_n \tilde{Z}) \quad \text{if } \tilde{A} = \mathbb{J}_n A \mathbb{J}_n. \quad (25c)$$

Reduction to basic displacements. A first consequence of Theorem 1, when it comes to computing specified inverse generators, is that the nine possible displacements defined in (1) can be reduced to the three basic ones shown in (4).

First, it follows from (13a) and (25a) that the case $N = \mathbb{Z}_{n,\psi}$ reduces to the case $N = \mathbb{Z}_{n,\psi}^T$. Similarly, (13b) and (25b) imply that the case $M = \mathbb{Z}_{n,\varphi}^T$ reduces to the case $M = \mathbb{Z}_{n,\varphi}$. We thus are left with the four cases defined by

$$M \in \{\mathbb{D}(x), \mathbb{Z}_{n,\varphi}\} \quad \text{and} \quad N \in \{\mathbb{D}(y), \mathbb{Z}_{n,\psi}^T\}.$$

Using (9) allows to further reduce the case where $M = \mathbb{Z}_{n,\varphi}$ and $N = \mathbb{D}(y)$ to the case where $M = \mathbb{D}(y)$ and $N = \mathbb{Z}_{n,\varphi}^T$. Due to the nature of the transformations applied to the $n \times \alpha$ generators (sign changes, permutations), the three reductions done so far imply an extra cost of only $O(\alpha n)$ operations in $\mathbb{K}$.

To reach (4) it remains to zero out the scalars φ and ψ . This can be done without transforming A , but only its displacement: for example, by combining the obvious identity

$$\mathbb{Z}_{n,\varphi} = \mathbb{Z}_{n,0} + \varphi \mathbf{e}_{n,1} \mathbf{e}_{n,n}^T, \quad (26)$$

with $\nabla[\mathbb{D}(x), \mathbb{Z}_{n,\psi}^T](A) = GH^T$ and $\nabla[\mathbb{Z}_{n,\varphi}, \mathbb{Z}_{n,\psi}^T](A) = GH^T$, we arrive at, respectively,

$$\nabla[\mathbb{D}(x), \mathbb{Z}_{n,0}^T](A) = \tilde{G} \tilde{H}^T \quad (i)$$

with $\tilde{G} = [G | \mathbf{A} \mathbf{e}_{n,n}]$ and $\tilde{H} = [H | \psi \mathbf{e}_{n,1}]$, and

$$\nabla[\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^T](A) = \tilde{G} \tilde{H}^T \quad (ii)$$

with $\tilde{G} = [G | -\varphi \mathbf{e}_{n,1} | \mathbf{A} \mathbf{e}_{n,n}]$ and $\tilde{H} = [H | A^T \mathbf{e}_{n,n} | \psi \mathbf{e}_{n,1}]$. The last column or row of A needed to set up the matrices $\tilde{G}$ and $\tilde{H}$ can be computed in $O(\alpha M(n) \log(n))$ —case (i)— or $O(\alpha M(n))$ —case (ii)— field operations from the explicit bilinear expressions of A given in [19, Examples 4.4.4 and 4.4.6(d)]. Due to the shape of $\tilde{G}, \tilde{H}$ above, extracting the first α columns of $\tilde{Y} = A^{-1} \tilde{G}$ and $\tilde{Z} = A^{-T} \tilde{H}$ in time $O(\alpha n)$ then yields the desired specified inverse generator (Y, Z) .

Reduction to strong regularity. Theorem 1 further allows to restrict to matrices that are not only invertible but strongly regular. Strong regularity, which is needed to apply Theorem 2 recursively, is classically obtained by preconditioning A into $\tilde{A} = P_1 A P_2$ with two random structured matrices P_1 and P_2 (see [19, §5.6]). Thus, one may generate $\tilde{A}$

as in Lemma 1, then compute an associated specified generator $(\tilde{Y}, \tilde{Z})$ of its inverse, and finally recover via Theorem 1 a specified generator (Y, Z) of the inverse of A .

Let r_1 and r_2 be two random vectors in $\mathbb{K}^n$ and whose first entry equals 1. Then, applying the rules of [19, p. 167], possible preconditioners for each of the three basic displacements of (4) are as follows (with $\tilde{x}, \tilde{y}$ in $\mathbb{K}^n$ and such that $\tilde{x}_i \neq x_i$ and $\tilde{y}_i \neq y_i$ for all i):

M, N	P_1	P_2
$\mathbb{D}(x), \mathbb{D}(y)$	$\mathbb{C}(\tilde{x}, x) \mathbb{D}(r_1)$	$\mathbb{C}(y, \tilde{y}) \mathbb{D}(r_2)$
$\mathbb{D}(x), \mathbb{Z}_{n,0}^T$	$\mathbb{C}(\tilde{x}, x) \mathbb{D}(r_1)$	$\mathbb{L}(r_2)$
$\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^T$	$\mathbb{U}(r_1)$	$\mathbb{L}(r_2)$

For all these cases, one may check that the structure of A, P_1 , and P_2 allows to prepare $(\tilde{G}, \tilde{H})$ in Lemma 1 and to recover (Y, Z) in Theorem 1 in time $O(\alpha M(n))$ or $O(\alpha M(n) \log(n))$.

3.2 Recursive factorization formula

THEOREM 2. Let $A \in \mathbb{K}^{n \times n}$ be nonsingular and generated by G and H as in (2). Assume that A_{11} is nonsingular as well, that it is generated by G_{11} and H_{11} as in (15), and let

$$Y_{11} = -A_{11}^{-1} G_{11}, \quad Z_{11} = A_{11}^{-T} H_{11}.$$

Assume further that the Schur complement S of A_{11} in A is generated by G_S and H_S as in (17), and let

$$Y_S = -S^{-1} G_S, \quad Z_S = S^{-T} H_S.$$

Then the matrices Y and Z in (3) satisfy

$$Y = F \begin{bmatrix} Y_{11}^{\rightarrow \alpha} \\ Y_S^{\rightarrow \alpha} \end{bmatrix}, \quad Z = E^T \begin{bmatrix} Z_{11}^{\rightarrow \alpha} \\ Z_S^{\rightarrow \alpha} \end{bmatrix},$$

where E and F are the elimination matrices defined in (7).

PROOF. Using (5) and (8), we obtain

$$-A^{-1} G = F \begin{bmatrix} -A_{11}^{-1} G_1 \\ -S^{-1} (G_2 - A_{21} A_{11}^{-1} G_1) \end{bmatrix}.$$

It follows from (15a) and (17a) that $G_1 = G_{11}^{\rightarrow \alpha}$ and that $G_2 - A_{21} A_{11}^{-1} G_1 = G_S^{\rightarrow \alpha}$. The expression claimed for $Y = -A^{-1} G$ then follows from applying the rule $A(B^{\rightarrow \alpha}) = (AB)^{\rightarrow \alpha}$ twice, and from the definitions of Y_{11} and Y_S . The expression for Z can be obtained in a similar way, using (15b) and (17b). $\square$

A first consequence of this theorem is a “compressed” analogue of the classical recursive factorization formula (8):

$$Y Z^T = F \begin{bmatrix} Y_{11}^{\rightarrow \alpha} \\ Y_S^{\rightarrow \alpha} \end{bmatrix} \begin{bmatrix} Z_{11}^{\rightarrow \alpha} \\ Z_S^{\rightarrow \alpha} \end{bmatrix}^T E.$$

A second consequence of Theorem 2 is that, for A strongly regular, we immediately get a recursive algorithm *à la* MBA whose key steps are the computation of generators (G_{11}, H_{11}) and (G_S, H_S) :

Given a generator (G, H) of length α for A ,

- Compute a generator (G_{11}, H_{11}) for A_{11} using (15);
- Recursively, compute $(Y_{11}, Z_{11}) = (-A_{11}^{-1} G_{11}, A_{11}^{-T} H_{11})$;
- Compute a generator (G_S, H_S) for S using (17);
- Recursively, compute $(Y_S, Z_S) = (-S^{-1} G_S, S^{-T} H_S)$;
- Compute $(-A^{-1} G, A^{-T} H)$ from the first α columns of Y_{11}, Y_S, Z_{11}, Z_S , using Theorem 2.

4. ALGORITHMS FOR LOWER TRIANGULAR OPERATOR MATRICES M AND N^T

In order to cover simultaneously the three displacements in (4) to which we have previously reduced, we assume in this section that both operator matrices M and N^T are lower triangular.

This assumption implies in particular that the blocks M₁₂ and N₂₁ in (6) are zero, so that their respective ranks μ and ν satisfy $\mu = \nu = 0$. From (15) it then follows that the submatrix A₁₁ satisfies

$$\nabla[M_{11}, N_{11}](A_{11}) = G_1 H_1^T. \quad (27)$$

Thus, some generators of length at most α for A₁₁ can be read off the first n_1 rows of some generators of length at most α for A.

Assuming that A₁₁ is invertible, consider now the associated specified generator of A₁₁⁻¹, that is,

$$Y_{11} = -A_{11}^{-1}G_1, \quad Z_{11} = A_{11}^{-T}H_1. \quad (28)$$

Combining the two identities in (28) with the explicit Schur complement generation formulas in (18) yields

$$\nabla[M_{22}, N_{22}](S) = (G_2 + A_{21}Y_{11})(H_2 - A_{12}^T Z_{11})^T. \quad (29)$$

In other words, the precise specification of the above generator of the inverse of A₁₁ can be exploited to simplify even further the generator of the Schur complement. In [4, Proposition 1], Cardinal had already noted this formula but only for the Cauchy-like structure (M and N diagonal).

Now, assuming further that A is strongly regular (which, if randomization is allowed, makes sense in view of the probabilistic reductions to strong regularity shown in Section 3.1), we obtain the following general algorithm:

GenInvLT(M, N, G, H)

Input: M, N $\in \mathbb{K}^{n \times n}$ and G, H $\in \mathbb{K}^{n \times \alpha}$ such that M and N^T are lower triangular, and $\nabla[M, N](A) = GH^T$.

Assumptions: A strongly regular, $m_{ii} \neq n_{jj}$ for all (i, j) .

Output: Y = -A⁻¹G and Z = A^{-T}H.

if $n = 1$ then

 Evaluate the dot product GH^T ;

 Deduce the scalar A;

 Y := -A⁻¹G; Z := A^{-T}H;

else

$n_1 := \lceil n/2 \rceil$; $n_2 := \lfloor n/2 \rfloor$;

 G₁₁ := G₁; H₁₁ := H₁;

 (Y₁₁, Z₁₁) := GenInvLT(M₁₁, N₁₁, G₁₁, H₁₁);

 G_S := G₂ + A₂₁Y₁₁; H_S := H₂ - A₁₂^TZ₁₁;

 (Y_S, Z_S) := GenInvLT(M₂₂, N₂₂, G_S, H_S);

 Y := $\begin{bmatrix} Y_{11} - A_{11}^{-1}A_{12}Y_S \\ Y_S \end{bmatrix}$; Z := $\begin{bmatrix} Z_{11} - A_{11}^{-T}A_{21}^TZ_S \\ Z_S \end{bmatrix}$;

fi;

return (Y, Z).

THEOREM 3. *Algorithm GenInvLT is correct.*

PROOF. When $n = 1$, the assumption on M and N implies that A is the scalar $(\sum_{i=1}^{\alpha} g_i h_{1i})/(m_{11} - n_{11})$. Correctness then follows immediately in this case. Assume now that $n > 1$ and, in order to proceed by induction, assume correctness for $n' < n$. The matrix A₁₁ is strongly regular (since A is) and it satisfies (27), where, by assumption M₁₁ and N₁₁^T are both lower triangular and with disjoint diagonals. Since $n_1 < n$, the induction assumption then implies

that the pair (Y₁₁, Z₁₁) returned by the first recursive call is precisely $(-A_{11}^{-1}G_1, A_{11}^{-T}H_1)$. Therefore, the computed pair (G_S, H_S) satisfies (29), where, by assumption, S is strongly regular (since A is) and where M₂₂ and N₂₂^T are both lower triangular and have disjoint diagonals. Since $n_2 < n$, the induction assumption implies that the pair (Y_S, Z_S) returned by the second recursive call is exactly $(-S^{-1}G_S, S^{-T}H_S)$. The conclusion then follows from Theorem 2. $\square$

To implement Algorithm GenInvLT and bound its cost, all we need is to be able to evaluate the four matrix products

$$A_{21}Y_{11}, \quad A_{12}^TZ_{11}, \quad A_{11}^{-1}A_{12}Y_S, \quad A_{11}^{-T}A_{21}^TZ_S. \quad (30)$$

In the next subsections, we study the evaluation of those expressions for each of three basic structures of the Cauchy, Vandermonde, and Hankel types. That requires in each case a detailed analysis of the structure of the matrices A₁₁⁻¹, A₁₂, A₂₁, and their transposes. Since in (30) there are two ways of parenthesizing the products of three matrices, we will also study the structure of A₁₁⁻¹A₁₂ and (A₂₁A₁₁⁻¹)^T. The parenthesizations (A₁₁⁻¹A₁₂)Y_S and (A₂₁A₁₁⁻¹)^TZ_S will be referred to as “Cardinal’s trick” later on, as they have been initially used in [4] for the Cauchy-like case.

4.1 Application to Cauchy-like matrices

We consider here the specialization of Algorithm GenInvLT to the Cauchy-like structure defined in (19a). Partitioning the two vectors x and y conformally with A yields

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad x_1, y_1 \in \mathbb{K}^{n_1}, \quad x_2, y_2 \in \mathbb{K}^{n_2}.$$

LEMMA 3. *Let the matrices A, G, H, Y₁₁, Z₁₁, G_S, H_S be as in Algorithm GenInvLT. Then*

- $\nabla[\mathbb{D}(x_i), \mathbb{D}(y_j)](A_{ij}) = G_i H_j^T$ for $1 \leq i, j \leq 2$,
- $\nabla[\mathbb{D}(y_1), \mathbb{D}(x_1)](A_{11}^{-1}) = Y_{11} Z_{11}^T$,
- $\nabla[\mathbb{D}(y_1), \mathbb{D}(y_2)](A_{11}^{-1}A_{12}) = -Y_{11} H_S^T$,
- $\nabla[\mathbb{D}(x_2), \mathbb{D}(x_1)](A_{21}A_{11}^{-1}) = G_S Z_{11}^T$.

PROOF. Since $\mathbb{D}(x)$ and $\mathbb{D}(y)$ are diagonal matrices, their off-diagonal blocks are zero, and the first identity follows from (2). To obtain the second identity, it suffices to pre- and postmultiply by A₁₁⁻¹ both sides of the first identity for $(i, j) = (1, 1)$, and then to use the specification of Y₁₁ and Z₁₁. Using the multiplication rule (10), we deduce further from the first identity for $(i, j) = (1, 2)$ and from the second one that

$$\begin{aligned} \nabla[\mathbb{D}(y_1), \mathbb{D}(y_2)](A_{11}^{-1}A_{12}) &= Y_{11} Z_{11}^T A_{12} + A_{11}^{-1} G_1 H_2^T \\ &= Y_{11} (Z_{11}^T A_{12} - H_2^T), \end{aligned}$$

which by definition of H_S equals $-Y_{11} H_S^T$. Similarly,

$$\begin{aligned} \nabla[\mathbb{D}(x_2), \mathbb{D}(x_1)](A_{21}A_{11}^{-1}) &= G_2 H_1^T A_{11}^{-1} + A_{21} Y_{11} Z_{11}^T \\ &= (G_2 + A_{21} Y_{11}) Z_{11}^T, \end{aligned}$$

which by definition of G_S equals $G_S Z_{11}^T$. $\square$

THEOREM 4. *Let n be a power of two and M, N $\in \mathbb{K}^{n \times n}$ be as in (19a). Then Algorithm GenInvLT requires at most*

$$3 \log(n) \text{MM}_{\mathbb{C}}(\alpha, n) + O(\alpha n \log(n))$$

field operations. If the set $\{x_1, \dots, x_n, y_1, \dots, y_n\}$ has cardinality $2n$ then this bound drops to

$$2 \log(n) \text{MM}_C(\alpha, n) + O(\alpha n \log(n)).$$

PROOF. When $n = 1$, $A = (\sum_{i=1}^{\alpha} g_i h_i) / (m_{11} - n_{11})$. Hence A^{-1} can be computed using $2\alpha + 1$ operations in $\mathbb{K}$, and the cost for $n = 1$ is $C(\alpha, 1) := 4\alpha + 2$. Consider now the case $n \geq 2$. Using Lemma 3 together with (9), we see that the matrices A_{11}^{-1} , A_{12} , A_{21} , and their transposes are all of the Cauchy-like structure defined in (19a). Furthermore, for each of them a generator of length at most α can be deduced in time $O(\alpha n)$ from the quantities computed by Algorithm GenInvLT. Consequently, one can compute $A_{21}Y_{11}$, $A_{12}^T Z_{11}$, $A_{11}^{-1}(A_{12}Y_5)$, and $A_{11}^{-T}(A_{21}^T Z_5)$ via six applications, in dimension $n/2$, of the reconstruction formula (19b) to α vectors in $\mathbb{K}^{n/2}$. Finally, Algorithm GenInvLT uses $2\alpha n$ additions to deduce G_5 , H_5 , and the upper parts of Y and Z . Overall, the cost for $n \geq 2$ thus satisfies

$$C(\alpha, n) \leq 2C(\alpha, n/2) + 6 \text{MM}_C(\alpha, n/2) + k\alpha n$$

for some constant k . The superlinearity of $\text{MM}_C(\cdot, n)$ then yields our first bound.

Assume now that the x_i and y_i are $2n$ pairwise distinct values. From Lemma 3 the reconstruction formula (19b) can then be applied directly to $A_{11}^{-1}A_{12}$ and to the transpose of $A_{21}A_{11}^{-1}$, in order to compute $(A_{11}^{-1}A_{12})Y_5$ and $(A_{21}A_{11}^{-1})^T Z_5$. This reduces the number of reconstructions from six to four, whence the second cost bound. $\square$

4.2 Application to Vandermonde-like matrices

Let us now focus on the cost of Algorithm GenInvLT when M and N correspond to the Vandermonde-like structure defined in (20a). We assume x to be partitioned as in the previous subsection.

LEMMA 4. Let the matrices $A, G, H, Y_{11}, Z_{11}, G_5, H_5$ be as in Algorithm GenInvLT. Let also w_{11} be the last column of A_{11} and v_{12}^T be the first row of $A_{11}^{-1}A_{12}$. Then

- $\nabla[\mathbb{D}(x_1), \mathbb{Z}_{n_2,0}^T](A_{12}) = G_1 H_2^T + w_{11} e_{n_2,1}^T$,
- $\nabla[\mathbb{D}(x_2), \mathbb{Z}_{n_1,0}^T](A_{21}) = G_2 H_1^T$,
- $\nabla[\mathbb{Z}_{n_1,0}^T, \mathbb{D}(x_1)](A_{11}^{-1}) = Y_{11} Z_{11}^T$,
- $\nabla[\mathbb{D}(x_2), \mathbb{D}(x_1)](A_{21}A_{11}^{-1}) = G_5 Z_{11}^T$,
- $\nabla[\mathbb{Z}_{n_1,1}^T, \mathbb{Z}_{n_2,0}^T](A_{11}^{-1}A_{12}) = -Y_{11} H_5^T + e_{n_1,n_1}(e_{n_2,1} + v_{12})^T$.

PROOF. In this case, the upper-right block of N satisfies $N_{12} = e_{n_1,n_1} e_{n_2,1}^T$. Hence we deduce from (2) that $\nabla[\mathbb{D}(x_1), \mathbb{Z}_{n_2,0}^T](A_{12}) = G_1 H_2^T + A_{11} e_{n_1,n_1} e_{n_2,1}^T$ and the first identity follows from the definition of vector w_{11} . The second to fourth identities are obtained in the same way as in the proof of Lemma 3. Let us now verify the last identity, which displays the structure of the product $A_{11}^{-1}A_{12}$. First, applying the techniques of Lemma 3, we deduce that

$$\nabla[\mathbb{Z}_{n_1,0}^T, \mathbb{Z}_{n_2,0}^T](A_{11}^{-1}A_{12}) = -Y_{11} H_5^T + e_{n_1,n_1} e_{n_2,1}^T.$$

Then, using (26) with $(\varphi, n) = (1, n_1)$ together with the definition of v_{12} yields the announced expression. $\square$

THEOREM 5. Let n be a power of two and $M, N \in \mathbb{K}^{n \times n}$ be as in (20a). Then Algorithm GenInvLT requires at most

$$3 \log(n) \text{MM}_V(\alpha, n) + O(\alpha M(n) \log^2(n))$$

field operations. If, in addition, the set $\{x_1, \dots, x_n\}$ has cardinality n then this bound drops to

$$2 \log(n) \text{MM}_V(\alpha, n) + O(\alpha M(n) \log^2(n)).$$

PROOF. When $n = 1$, $A^{-1} = x_1 / (\sum_{i=1}^{\alpha} g_i h_i)$, so that the cost is $C(\alpha, 1) := 4\alpha + 1$. Assume now that $n \geq 2$. Lemma 4 implies that A_{12} , A_{21} , and A_{11}^{-T} share the same Vandermonde-like structure (20a) as A and A_{11} . However, A_{12} has displacement rank bounded by $\alpha + 1$ and computing its generator can be done at cost $O(\alpha M(n) \log(n))$ by applying (20b) to A_{11} . Hence, for $n \geq 2$,

$$C(\alpha, n) \leq 2C(\alpha, n/2) + 4 \text{MM}_V(\alpha, n/2) + 2 \text{MM}_V(\alpha + 1, n/2, \alpha) + k\alpha M(n) \log(n),$$

for some constant k . From (22a), and the superlinearity of $M(n)$ and $\text{MM}_V(\cdot, n)$, we then deduce the first cost bound.

If all the x_i are distinct then, for $A_{21}A_{11}^{-1}$, we proceed as for the Cauchy-like case. For $A_{11}^{-1}A_{12}$, note that $\mathbb{J}_{n_1} A_{11}^{-1} A_{12}$ is Hankel-like in the sense of (21a). Hence, one may first generate the latter matrix in time $O(\alpha M(n) \log(n))$ by obtaining the vector v_{12} after two applications of (20b), then multiply by Y_5 using (21b), and re-apply a reflexion. Thus,

$$C(\alpha, n) \leq 2C(\alpha, n/2) + \text{MM}_V(\alpha, n/2) + \text{MM}_V(\alpha + 1, n/2, \alpha) + \text{MM}_C(\alpha, n/2) + \text{MM}_H(\alpha + 1, n/2, \alpha) + k\alpha M(n) \log(n),$$

for some constant k , and the conclusion follows as before. $\square$

Note that unlike for the Cauchy-like case, if α is small enough then in the cost bounds of Theorem 5 both summands have the same order of magnitude.

4.3 Extension to Hankel-like matrices

Finally, let us consider the Hankel-like structure defined by $M = \mathbb{Z}_{n,0}$ and $N = \mathbb{Z}_{n,0}^T$. Although M and N^T are lower triangular, Algorithm GenInvLT cannot be used directly in this case as the operator $\nabla[\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^T]$ is not invertible. Covering such a structure, however, is interesting in particular as it yields an immediate extension to some Toeplitz-like matrices (see [19, Remark 5.4.4] and our Section 3.1).

To cope with the singularity of the displacement operator, some additional data, called *irregularity set* in [19, p. 136], are needed, which typically consist in “a few” entries of A . An irregularity set for $\nabla[\mathbb{Z}_{n,0}, \mathbb{Z}_{m,0}^T]$ is given by the last row of A . Indeed, for $u^T = e_{n,n}^T A$ we see that (2) and (26) imply

$$\nabla[\mathbb{Z}_{n,1}, \mathbb{Z}_{n,0}^T](A) = [G \mid e_{n,1}] [H \mid u]^T, \quad (31)$$

so that the matrix A is Hankel-like in the sense of (21a), with displacement rank $\alpha + 1$. Consequently, the reconstruction formula (21b) can be used.

We need to exhibit an irregularity set for $\nabla[\mathbb{Z}_{n,0}^T, \mathbb{Z}_{n,0}]$ too, because we shall multiply with inverses of Hankel-like matrices. A suitable choice here is $v^T = e_{n,1}^T A^{-1}$, the first row of the inverse of A : indeed, if $\nabla[\mathbb{Z}_{n,0}^T, \mathbb{Z}_{n,0}](A^{-1}) = YZ^T$ then, recalling (13c), we may check that $\mathbb{J}_n A^{-1} \mathbb{J}_n$ satisfies an identity similar to (31); it is thus fully determined by, up to reflexions, Y , Z , and its last row $v^T \mathbb{J}_n$.

The resulting adaptation of Algorithm **GenInvLT** to the Hankel-like operator $\nabla[\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^T]$ is as follows:

GenInvHL(G, H, u)

Input: $G, H \in \mathbb{K}^{n \times \alpha}$ such that $\nabla[\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^T](A) = GH^T$,
and $u = A^T e_{n,n}$ (the last row of A).
Assumption: A strongly regular.
Output: $Y = -A^{-1}G$, $Z = A^{-T}H$, and $v = A^{-T}e_{n,1}$
(the first row of A^{-1}).

if $n = 1$ then
 $Y := -u^{-1}G$; $Z := u^{-1}H$; $v := u^{-1}$;
else
 $n_1 := \lceil n/2 \rceil$; $n_2 := \lfloor n/2 \rfloor$;
 $\begin{bmatrix} u_{11} \\ u_{12} \end{bmatrix} := A^T e_{n,n_1}$; $\begin{bmatrix} u_{21} \\ u_{22} \end{bmatrix} := u$;
 $(Y_{11}, Z_{11}, v_{11}) := \text{GenInvHL}(G_1, H_1, u_{11})$;
 $G_S := G_2 + A_{21}Y_{11}$; $H_S := H_2 - A_{12}^T Z_{11}$;
 $u_S := u_{22} - A_{12}^T A_{11}^{-T} u_{21}$;
 $(Y_S, Z_S, v_S) := \text{GenInvHL}(G_S, H_S, u_S)$;
 $Y := \begin{bmatrix} Y_{11} - (A_{11}^{-1} A_{12}) Y_S \\ Y_S \end{bmatrix}$; $Z := \begin{bmatrix} Z_{11} - (A_{11}^{-T} A_{21}^T) Z_S \\ Z_S \end{bmatrix}$;
 $w := -S^{-T} A_{12}^T v_{11}$; $v := \begin{bmatrix} v_{11} - A_{11}^{-T} A_{21}^T w \\ v_S \end{bmatrix}$;
fi;
return (Y, Z, v) .

THEOREM 6. *Algorithm GenInvHL is correct.*

PROOF. When $n = 1$, both A and u are reduced to the scalar $a_{1,1}$ and correctness is then straightforward. Assume now that $n > 1$ and, in order to proceed by induction, assume correctness for $n' < n$. The vector u is split into $u_{21} \in \mathbb{K}^{n_1}$ and $u_{22} \in \mathbb{K}^{n_2}$. Similarly, the vector of coefficients of row n_1 of A is split into $u_{11} \in \mathbb{K}^{n_1}$ and $u_{12} \in \mathbb{K}^{n_2}$. Hence u_{11} equals $A_{11}^T e_{n_1, n_1}$ (that is, the vector of coefficients of the last row of A_{11}), $u_{22} = A_{22}^T e_{n_2, n_2}$, and $u_{21} = A_{21}^T e_{n_2, n_2}$. Recalling that $S = A_{22} - A_{21} A_{11}^{-1} A_{12}$, we deduce that the vector u_S computed by Algorithm **GenInvHL** satisfies $u_S = S^T e_{n_2, n_2}$ and thus is the vector of coefficients of the last row of S . Since the computation of Y and Z is unchanged in comparison to Algorithm **GenInvLT**, we still have $Y = -A^{-1}G$ and $Z = A^{-T}H$. All that remains is to prove that v is actually the vector of coefficients of the first row of A^{-1} . By induction, v_{11}^T and v_S^T correspond to the first rows of A_{11}^{-1} and S^{-1} , respectively. Using the factorization of A^{-1} seen in (8) and letting $w^T = -v_{11}^T A_{12} S^{-1}$, we get:

$$\begin{aligned} e_{n,1}^T A^{-1} &= \begin{bmatrix} e_{n_1,1}^T & -e_{n_1,1}^T A_{11}^{-1} A_{12} \end{bmatrix} \begin{bmatrix} A_{11}^{-1} \\ S^{-1} \end{bmatrix} E \\ &= \begin{bmatrix} v_{11}^T & w^T \end{bmatrix} E \\ &= \begin{bmatrix} v_{11}^T - w^T A_{21} A_{11}^{-1} & w^T \end{bmatrix}, \end{aligned}$$

which is exactly the way vector v is computed. $\square$

LEMMA 5. *Let $A, G, H, Y_{11}, Z_{11}, G_S, H_S, u_{11}$ be as in Algorithm GenInvHL. Recall that u_{11} is the last row of the matrix A_{11} and let w_{11} be its last column. Then*

- $\nabla[\mathbb{Z}_{n_1,0}, \mathbb{Z}_{n_2,0}^T](A_{12}) = G_1 H_2^T + w_{11} e_{n_2,1}^T$,
- $\nabla[\mathbb{Z}_{n_2,0}, \mathbb{Z}_{n_1,0}^T](A_{21}) = G_2 H_1^T - e_{n_2,1} u_{11}^T$,
- $\nabla[\mathbb{Z}_{n_1,0}^T, \mathbb{Z}_{n_1,0}](A_{11}^{-1}) = Y_{11} Z_{11}^T$,
- $\nabla[\mathbb{Z}_{n_1,0}^T, \mathbb{Z}_{n_2,0}^T](A_{11}^{-1} A_{12}) = -Y_{11} H_S^T + e_{n_1, n_1} e_{n_2,1}^T$,

- $\nabla[\mathbb{Z}_{n_2,0}, \mathbb{Z}_{n_1,0}](A_{21} A_{11}^{-1}) = G_S Z_{11}^T - e_{n_2,1} e_{n_1, n_1}^T$.

PROOF. Proceed as for Lemma 3 and Lemma 4. $\square$

THEOREM 7. *Let n be a power of two and $M, N \in \mathbb{K}^{n \times n}$ be as in (21a). Then Algorithm GenInvHL requires at most*

$$2 \log(n) \text{MM}_H(\alpha, n) + O(\alpha M(n) \log(n)).$$

field operations.

PROOF. When $n = 1$, u is a scalar and the algorithm has cost $C(\alpha, 1) := 2\alpha + 2$. Assume now $n \geq 2$. Given G, H , and u , one has (31) and thus (21b) yields $[u_{11}^T, u_{12}^T]$ in time $O(\alpha M(n))$. From Lemma 5, all the blocks involved have the same structure as A , up to transposition and row/column reflexion, and with sometimes a displacement rank $\alpha + 1$ instead of α . Generating these blocks requires the knowledge of the vectors u_{11} (already computed) and w_{11} (computable as u_{11}), which has cost $O(\alpha M(n))$. Now, one may check that the irregularity sets of $A_{12}, A_{21}, \mathbb{J}_{n_1} A_{11}^{-1} \mathbb{J}_{n_1}, \mathbb{J}_{n_1} A_{11}^{-1} A_{12}, A_{21} A_{11}^{-1} \mathbb{J}_{n_1}$, and $\mathbb{J}_{n_2} S^{-1} \mathbb{J}_{n_2}$ are, respectively, $u_{12}, u_{21}, v_{11}, v_{11}^T A_{12}, u_{21} A_{11}^{-1} \mathbb{J}_{n_1}$, and v_S . The vector u_{12} has already been computed, u_{21} is part of the input, v_{11} and v_S are computed recursively, and the two remaining vectors can be recovered in time $O(\alpha M(n))$ from u_{21}, v_{11} , and the generators of A_{11}^{-1} and A_{12} . Consequently, all the products that appear in Algorithm **GenInvHL** can be produced by applications of (21b). Finally, Algorithm **GenInvHL** still uses $O(\alpha n)$ additions, so that the total cost bound is given by

$$C(\alpha, n) \leq 2C(\alpha, n/2) + 4 \text{MM}_H(\alpha + 1, n/2, \alpha) + k \alpha M(n),$$

for some constant k . The conclusion follows from (22b) and the superlinearity assumptions. $\square$

5. EXPERIMENTAL RESULTS

We have implemented the two variants of **GenInvLT** (with and without Cardinal's trick) as well as the MBA algorithm for Sylvester's displacement. Moreover, we have developed some code to handle Cauchy-like and Hankel-like structures.

For our experiments, we take $\mathbb{K} = \mathbb{F}_p$ with $p = 999999937$, which lets us measure the algebraic costs. Basic operations in $\mathbb{K}$ are provided by NTL,² and we also use some code for fast polynomial arithmetic.³ All the computations are carried out on a desktop machine with an Intel® Core™ 2 Duo processor at 2.66 GHz. Finally, generators (G, H) are picked randomly, while operator matrices $\mathbb{D}(x), \mathbb{D}(y)$ are chosen in order to satisfy all the assumptions made on the algorithms.

Figure 1 shows computing times for inverting Cauchy-like matrices of displacement rank $\alpha = 10$ when n is increasing. It appears that the computing time is quasi-linear with respect to n for each method, and that the compression steps in MBA have negligible cost. Thus, the main difference explaining the various performances lies in the number of products “Cauchy-like matrix $\times$ vectors.” We have already seen in Theorem 4 that the choice in the parenthesizations leads to one variant in $3 \log(n) \text{MM}_C(\alpha, n)$ and, up to stronger conditions on the input, to another variant in $2 \log(n) \text{MM}_C(\alpha, n)$. Let us now estimate this cost for our implementation of the MBA algorithm. Generators for the Schur complement and the inverse of A before the compression steps are computed using (10) according to the following parenthesization: $X_1 = A_{11}^{-1} A_{12}$, $S = A_{22} - A_{21} X_1$,

²<http://www.shoup.net/ntl/>

³<http://www.math.uvsq.fr/~lecerf/software/tellegen/>

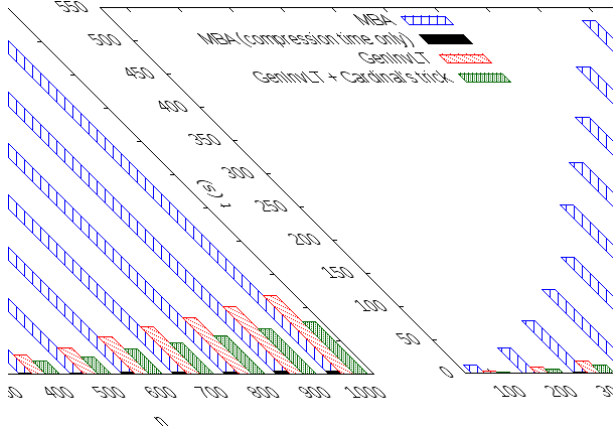


Figure 1: Cost (in seconds) of Cauchy-like matrix inversion for $\alpha = 10$ and increasing values of n .

$X_2 = A_{21}A_{11}^{-1}$, and

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} + (X_1 S^{-1})X_2 & -X_1 S^{-1} \\ -S^{-1}X_2 & S^{-1} \end{bmatrix}.$$

Counting the costs of all these products using (23) and the superlinearity of $MM_C(\cdot, n)$ leads to a bound of $14 MM_C(\alpha, n)$ in the recurrence equation for the cost of MBA, which gives a total cost dominated by $14 \log(n) MM_C(\alpha, n)$. In Figure 1, we observe a speed-up around $4.6 \approx 14/3$ between MBA and our first variant (GenInvLT), and around $6.7 \approx 14/2$ between MBA and the second variant (GenInvLT + Cardinal's trick), which is in agreement with our analysis above.

Moreover, we experimented with Hankel-like matrices in order to estimate the cost of row reconstruction and the additional time due to subblocks having displacement rank $\alpha + 1$ instead of α like in the Cauchy-like case. Timings are summarized in Table 1, where it appears that these costs become negligible when α is large enough. Indeed, they are linear in α as expected (see (22b) and Theorem 7) while the total cost seems quadratic in α .

α	10	30	50	70	90
Total cost	4.7	34.7	92.1	177.0	290.3
Irregularity related cost	0.8	2.5	3.9	5.3	7.1
Rank-increase related cost	0.5	1.6	2.6	3.5	4.6

Table 1: Cost (in seconds) of Hankel-like matrix inversion for $n = 200$ and increasing values of α .

6. CONCLUSIONS

In this paper, we have extended Cardinal's compression-free algorithm to a broader class of structured matrices, including not only the Cauchy-like type but also the Vandermonde-, Hankel-, and Toeplitz-like types. Our main conclusion is that this approach yields variants of the MBA algorithm that are simpler to analyze and implement, and, according to our first experiments, significantly faster in practice. However, this study calls for a number of extensions:

On the practical side, we should first study the impact of stopping recursive calls (and reconstructing A^{-1} explicitly via fast dense linear algebra) when $n \approx \alpha$. It would also be interesting to measure the memory gains brought by Cardinal's extended approach over MBA.

On the algorithmic side, although we have focused only on $O(\alpha^2 n)$ versions of MBA, it would be interesting to incorporate the matrix multiplication techniques of [3]. We

should also study the impact of multiplicities in x and y on the cost bounds and adapt our work to structures like those of the Toeplitz+Hankel-like type.

7. ACKNOWLEDGMENTS

We thank Benoît Lacelle for providing a framework for structured matrices and the generator-compression subroutine used in our implementation of MBA, Éric Schost for pointing out the code used for fast multipoint evaluation, and Gilles Villard for useful discussions.

8. REFERENCES

- [1] R. R. Bitmead and B. D. O. Anderson. Asymptotically fast solution of Toeplitz and related systems of linear equations. *Linear Algebra Appl.*, 34:103–116, 1980.
- [2] A. Bostan, C.-P. Jeannerod, and É. Schost. Solving Toeplitz- and Vandermonde-like linear systems with large displacement rank. In *ISSAC'07*, pages 33–40. ACM, 2007.
- [3] A. Bostan, C.-P. Jeannerod, and É. Schost. Solving structured linear systems with large displacement rank. *Theoretical Computer Science*, 407(1:3):155–181, 2008.
- [4] J.-P. Cardinal. On a property of Cauchy-like matrices. *C. R. Acad. Sci. Paris - Série I - Analyse numérique/Numerical Analysis*, 328:1089–1093, 1999.
- [5] J.-P. Cardinal. A divide and conquer method to solve Cauchy-like systems. Technical report, The FRISCO Consortium, 2000.
- [6] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, second edition, 2003.
- [7] I. Gohberg and V. Olshevsky. Complexity of multiplication with vectors for structured matrices. *Linear Algebra Appl.*, 202:163–192, 1994.
- [8] I. Gohberg and V. Olshevsky. Fast state space algorithms for matrix Nehari and Nehari-Takagi interpolation problems. *Integral Equations and Operator Theory*, 20:44–83, 1994.
- [9] G. Heinig. Inversion of generalized Cauchy matrices and other classes of structured matrices. *Linear Algebra for Signal Processing*, 69:95–114, 1995.
- [10] T. Kailath, S. Y. Kung, and M. Morf. Displacement ranks of matrices and linear equations. *J. Math. Anal. Appl.*, 68(2):395–407, 1979.
- [11] E. Kaltofen. Asymptotically fast solution of Toeplitz-like singular linear systems. In *ISSAC'94*, pages 297–304. ACM, 1994.
- [12] E. Kaltofen. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. *Mathematics of Computation*, 64(210):777–806, 1995.
- [13] M. Morf. Doubling algorithms for Toeplitz and related equations. *IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 954–959, 1980.
- [14] V. Olshevsky and V. Pan. A unified superfast algorithm for boundary rational tangential interpolation problems and for inversion and factorization of dense structured matrices. In *Proc. 39th IEEE FOCS*, pages 192–201, 1998.
- [15] V. Y. Pan. Parallel solution of Toeplitz-like linear systems. *Journal of Complexity*, 8(1):1–21, 1992.
- [16] V. Y. Pan. Parametrization of Newton's iteration for computations with structured matrices and applications. *Computers Math. Applic.*, 24(3):61–75, 1992.
- [17] V. Y. Pan. Decreasing the displacement rank of a matrix. *SIAM J. Matrix Anal. Appl.*, 14(1):118–121, 1993.
- [18] V. Y. Pan. Nearly optimal computations with structured matrices. In *SODA'00*, pages 953–962. ACM, 2000.
- [19] V. Y. Pan. *Structured Matrices and Polynomials*. Birkhäuser Boston Inc., 2001.
- [20] V. Y. Pan and A. Zheng. Superfast algorithms for Cauchy-like matrix computations and extensions. *Linear Algebra Appl.*, 310:83–108, 2000.

Yet Another Block Lanczos Algorithm: How To Simplify the Computation and Reduce Reliance on Preconditioners in the Small Field Case

Wayne Eberly^{*}

Department of Computer Science
University of Calgary
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
eberly@ucalgary.ca

ABSTRACT

A new block Lanczos algorithm for computations over small finite fields is presented and analysed. The algorithm can be used to solve a system of linear equations or sample uniformly from the null space whenever the number of nilpotent blocks with order at least two in the Jordan form of the given coefficient matrix is less than the block factor on the right. It can also be used to verify that this matrix condition is *not* satisfied, in order to confirm that preconditioning of the given matrix is required.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—*algebraic algorithms, analysis of algorithms*; F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—*computations in finite fields, computations in matrices*

General Terms

Algorithms, Reliability, Performance

Keywords

Randomized computations, computations over small fields, Lanczos algorithms

1. INTRODUCTION

Since the mid nineteen-eighties, Krylov-based algorithms have been used to solve systems of linear equations over finite fields or to sample from the null space of matrices over such fields, as needed to solve a variety of problems. A considerable amount of work has subsequently taken place to improve

the efficiency and reliability of these methods; the LinBox home page, www.linalg.org, is a good source for additional references about this. These techniques have been effective when storage requirements prohibit the use of elimination-based methods and when other special-purpose techniques have not been available.

Various matrix properties have been assumed when proving the reliability of these methods. For computations over large fields these assumptions have not been problematic, because extremely simple and efficient matrix “preconditioners” can be used to establish the properties that are required; quite a few of these are presented in the report of Chen et al. [2]. Unfortunately the set of preconditioners available for computations over small fields is more limited – to my knowledge only a sparse preconditioner first described by Wiedemann has presently been analyzed (see [10], [2], [4]), and this is somewhat more costly than desirable.

At present, the set of problems that can be solved reliably over small fields by these techniques without preconditioning is quite limited: Villard [9] has demonstrated that if rectangular blocks are used (with the block size on the left exceeding the block size on the right by at least two) then a block Wiedemann algorithm can be used, reliably, to find a nonzero element of the null space of any singular matrix.

In this paper two additional matrix problems are considered, namely, the solution of a linear system $Ax = b$ (returning either a solution for the system or a certificate that it is inconsistent) and the problem of the uniform and random selection of elements from the null space — a problem that must be solved for computations over F_2 when sieve-based algorithms for integer factorization are applied [1].

As their names suggest, Krylov-based algorithms perform computations over the *Krylov space* of a set of vectors. In particular, an algorithm using block size r on the right requires that an initial set of r vectors $v_1, \dots, v_r$ is somehow provided, and the algorithm (either implicitly or explicitly) carries out a search in the Krylov space generated by these vectors, that is, the space spanned by the set of vectors $A^i v_j$ for $i \geq 0$ and $1 \leq j \leq r$. Virtually all of the Krylov-based algorithms that have been investigated to date require either that the vectors $v_1, \dots, v_r$ are given as part of the input or that they are randomly generated.

Suppose now that k is a positive integer. We will say that a matrix A (with entries in a finite field F_q with q elements) is *k-derogatory* if the first k invariant factors of the matrix are divisible by x^2 or, equivalently, the Jordan normal form

^{*}Research supported in part by Natural Sciences and Engineering Research Council of Canada grant OGP00089756.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010 Munich, Germany

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

of A includes at least k nilpotent Jordan blocks with order at least two. We will call the matrix A k -nonderogatory otherwise. If the above-mentioned vectors $v_1, \dots, v_r$ are to be chosen randomly then it is necessary for the coefficient matrix A to be nonderogatory if a Krylov-based algorithm is to be used reliably to solve a linear system. This is also a necessary condition if one wishes to sample uniformly and randomly from the null space of a given matrix.

In this paper a new block Lanczos algorithm is presented. A significant difference from previous methods described by Coppersmith [3] and Montgomery [8] is that the input matrix A is not required to be symmetric: Coppersmith's and Montgomery's algorithms each work with the matrix $A^t A$ whenever $A \neq A^t$. Unfortunately symmetrization of the input (at least, in this way) can be harmful for computations over finite fields instead of helpful: Consider, for example, the matrices

$$A_1 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \in \mathbb{F}_2^{2 \times 2}.$$

Since the system $A_1 x = b$ is consistent if and only if b is itself a solution, the Krylov space generated by b with respect to A_1 contains a solution for this system whenever a solution exists at all. On the other hand, $A_1^t A_1 = A_2$, and this matrix is nilpotent — $A_2^2 = 0$, so that the Krylov space generated by b with respect to A_2 only includes a solution for the system $A_2 x = b$ if $b = 0$. Worse yet, a consideration of block diagonal matrices A including A_2 on the diagonal confirms that the rank of the matrix $A^t A$ can be significantly different from that of A , so that symmetrization can also change the null space whose entries one wishes to sample.

The new algorithm differs from the block Lanczos algorithm of Hovinen [7] in its use of rectangular blocks, such that the difference between the block size on the left and the block size on the right is at least logarithmic in the order of the coefficient matrix. Using rectangular blocks one can show that, with high probability, the Krylov space on the right is completely searched during the computation at low cost. The orthogonalization process (that is at the heart of any “Lanczos” algorithm) used here is also somewhat simpler than that of Hovinen's algorithm.

While proofs of the efficiency and reliability of the algorithm are omitted in this abstract, a longer version of the paper that includes all proofs, as well as additional algorithmic details, is now available [5].

Henceforth $\mathbb{F}_q$ will denote the finite field with q elements and, for integers i and j , $\mathbb{F}_q^{i \times j}$ will denote the set of $i \times j$ matrices with entries in $\mathbb{F}_q$.

2. ITERATION OVER A KRYLOV SPACE

Given a matrix $A \in \mathbb{F}_q^{n \times n}$, r vectors $v_1, \dots, v_r \in \mathbb{F}_q^{n \times 1}$ and a positive integer δ , the algorithm described in this section will either traverse the Krylov space generated by $v_1, \dots, v_r$ or will fail — the latter happening with probability less than $7q^{-\delta}$.

The algorithm begins with a Lanczos phase and, if necessary, ends with an elimination phase, in order to generate the following sequences.

- S_{lan} includes m_{lan} ordered triples of vectors

$$S_{\text{lan}} = (\alpha_1, \beta_1, \gamma_1), \dots, (\alpha_{m_{\text{lan}}}, \beta_{m_{\text{lan}}}, \gamma_{m_{\text{lan}}}) \quad (1)$$

1. Set ℓ ; initialize $S_{\text{lan}}, S_{\text{null}}, S_{\text{elim}}, S_{\text{next}}, s_L, L_0$ and U
2. **while** (S_{next} is nonempty and S_{elim} is empty) **do**
3. **if** ($|U| < \ell - \lceil \log_q n \rceil - \delta$) **then**
4. **if** ($U \subseteq \{(s_L, 1), (s_L, 2), \dots, (s_L, \ell)\}$) **then**
5. $s_L := s_L + 1$; initialize the sequence L_{s_L}
6. $U := U \cup \{(s_L, 1), (s_L, 2), \dots, (s_L, \ell)\}$
7. **else**
8. Report failure and halt
9. **end if**
10. **end if**
11. Match vectors listed in U with vectors in S_{next} and extend the sequence S_{lan}
12. **if** (all vectors in S_{next} were matched) **then**
13. Update vectors listed in U
14. **else**
15. Update S_{elim} (which will now be nonempty)
16. **end if**
17. Update S_{next}
18. **end while**

Figure 1: Lanczos Phase of the Main Algorithm

such that, for $1 \leq i, j \leq m_{\text{lan}}, \alpha_i, \beta_i, \gamma_j \in \mathbb{F}_q^{n \times 1}$,

$$\beta_i = A\alpha_i \quad \text{and} \quad \gamma_i^T A\alpha_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \quad (2)$$

- S_{null} includes m_{null} linearly independent vectors

$$S_{\text{null}} = \delta_1, \dots, \delta_{m_{\text{null}}} \quad (3)$$

such that, for $1 \leq i \leq m_{\text{null}}, \delta_i \in \mathbb{F}_q^{n \times 1}$ and

$$A\delta_i = 0. \quad (4)$$

- S_{elim} includes m_{elim} ordered pairs of vectors

$$S_{\text{elim}} = (\epsilon_1, \zeta_1), \dots, (\epsilon_{m_{\text{elim}}}, \zeta_{m_{\text{elim}}}) \quad (5)$$

where $\epsilon_i, \zeta_i \in \mathbb{F}_q^{n \times 1}$, $\zeta_1, \dots, \zeta_{m_{\text{elim}}}$ are linearly independent and, for $1 \leq h \leq m_{\text{lan}}$ and $1 \leq i \leq m_{\text{elim}}$,

$$\zeta_i = A\epsilon_i \quad \text{and} \quad \gamma_h^t A\epsilon_i = 0. \quad (6)$$

In particular, the algorithm returns sequences $S_{\text{lan}}, S_{\text{null}}$ and S_{elim} , as given above, such that the vectors

$$\alpha_1, \dots, \alpha_{m_{\text{lan}}}, \delta_1, \dots, \delta_{m_{\text{null}}}, \epsilon_1, \dots, \epsilon_{m_{\text{elim}}}$$

form a basis for the Krylov space $\mathcal{K}$ generated by $v_1, \dots, v_r$.

The algorithm begins with a uniform and independent selection of vectors $u_1, \dots, u_\ell \in \mathbb{F}_q^{n \times 1}$ where

$$\ell \geq r + 2 (\lceil \log_q n \rceil + \delta). \quad (7)$$

All subsequent steps of this algorithm are deterministic.

2.1 Details of the Lanczos Phase

The major stages of the Lanczos phase are shown in Figure 1. Throughout its execution $S_{\text{lan}}, S_{\text{null}}$ and S_{elim} are sequences of vectors with lengths $m_{\text{lan}}, m_{\text{null}}$ and m_{elim} , respectively, satisfying the various conditions shown in Equations (1)–(6), above. The integer s_L represents the stage of the generation of vectors on the left currently in progress.

The algorithm maintains sequences $L_0, \dots, L_{s_L}$ of vectors, each of length ℓ : At each point in the computation

$$L_i = \eta_{i,1}, \dots, \eta_{i,\ell} \quad (8)$$

where $\eta_{i,j} \in \mathbb{F}_q^{n \times 1}$ for $0 \leq i \leq s_L$ and $1 \leq j \leq \ell$. Vectors in these sequences are either “processed” or “unprocessed.” A set of ordered pairs of integers

$$U \subseteq \{(i, j) \mid 0 \leq i \leq s_L \text{ and } 1 \leq j \leq \ell\} \quad (9)$$

is used to keep track of the unprocessed vectors: For $0 \leq i \leq s_L$ and $1 \leq j \leq \ell$, the j^{th} vector $\eta_{i,j}$ in the sequence L_i is unprocessed if and only if $(i, j) \in U$.

The algorithm also maintains a sequence S_{next} of ordered pairs of vectors

$$S_{\text{next}} = (\theta_1, \iota_1), \dots, (\theta_{m_{\text{next}}}, \iota_{m_{\text{next}}}) \quad (10)$$

where $\iota_i = A\theta_i$ for $1 \leq i \leq m_{\text{next}}$, to continue generation of the desired Krylov space, as described below.

2.1.1 Initialization of Sequences (Step 1)

The sequences S_{lan} , S_{null} and S_{elim} are initially empty. The sequence S_{next} is then initialized and S_{null} is updated by beginning with the sequence of ordered pairs

$$(v_1, Av_1), \dots, (v_r, Av_r) \quad (11)$$

and applying elimination on the second entries to detect linear dependencies; the first entries of ordered pairs are updated as second entries are. Following this process S_{null} and S_{next} are as shown at lines (3) and (10) respectively, $\iota_1, \dots, \iota_{m_{\text{next}}}$ are linearly independent, and the vectors

$$\delta_1, \dots, \delta_{m_{\text{null}}}, \theta_1, \dots, \theta_{m_{\text{next}}}$$

spans the same vector space as $v_1, \dots, v_r$.

The sequence L_0 is set to be

$$L_0 = u_1, \dots, u_\ell, \quad (12)$$

so that $\eta_{0,i} = u_i$ for $1 \leq i \leq \ell$ and, since each of these vectors is initially unprocessed,

$$U = \{(0, 1), \dots, (0, \ell)\}. \quad (13)$$

2.1.2 Initialization of L_{s_L} (Step 5)

A new sequence L_{s_L} of vectors is generated at step 5 whenever the number of unprocessed vectors falls below a certain threshold and provided that these vectors are all recent (see steps 3 and 4 in Figure 1). In particular, for $1 \leq j \leq \ell$, the vector $\eta_{s_L,j}$ is initialized to have value

$$\eta_{s_L,j} := A^t \eta_{s_L-1,j}, \quad (14)$$

where $\eta_{s_L-1,j}$ is j^{th} vector in the sequence L_{s_L-1} . However an update is immediately performed,

$$\eta_{s_L,j} := \eta_{s_L,j} - \sum_{h=\text{low}_L}^{m_{\text{lan}}} \eta_{s_L,j}^t \beta_h \cdot \gamma_h \quad (15)$$

where

$$\text{low}_L = \max(m_{\text{lan}} - 2\ell - 2r + 1, 1). \quad (16)$$

This will ensure that $\eta_{s_L,j}^t A \alpha_i = 0$ for $1 \leq i \leq m_{\text{lan}}$.

2.1.3 Matching Vectors in U and S_{next} (Step 8)

Let $m_{\text{avail}} = |U|$ and let $L \in \mathbb{F}_q^{n \times m_{\text{avail}}}$ and $R \in \mathbb{F}_q^{n \times m_{\text{next}}}$ be matrices whose columns are the vectors $\eta_{i,j}$ where $(i, j) \in U$, and the first entries of the ordered pairs in S_{next} , respectively. Vectors in U are matched with pairs in S_{next} by determining the rank m_{sel} of $L^T AR$, and by finding permutation matrices $P_L \in \mathbb{F}_q^{m_{\text{avail}} \times m_{\text{avail}}}$ and $P_R \in \mathbb{F}_q^{m_{\text{next}} \times m_{\text{next}}}$ such

that the leading $i \times i$ minors of the matrix $P_L L^T ARP_R$ are nonsingular for $1 \leq i \leq m_{\text{sel}}$: The vectors with indices in U and the pairs in S_{next} that have been “matched” are found in the top m_{sel} rows and columns of $P_L L^T ARP_R$.

While P_R can be chosen arbitrarily, the vectors $\eta_{i,j}$ should be used in a greedy way: The first m_{sel} columns of LP_L^t should be vectors $\eta_{i_1,j_1}, \dots, \eta_{i_{m_{\text{sel}}},j_{m_{\text{sel}}}}$, where

$$(i_1, j_1), \dots, (i_{m_{\text{sel}}}, j_{m_{\text{sel}}})$$

is the lexicographically first sequence of m_{sel} distinct ordered pairs of U such that the leading minors of the corresponding matrix $P_L^t L ARP_R$ are nonsingular.

Since the matrix $L^t AR$ is small (it has fewer than 2ℓ rows and at most r columns), Gaussian elimination can be used to perform the above matching.

Let m_{old} be the length of the sequence S_{lan} before this step and, for $1 \leq h \leq m_{\text{sel}}$, suppose that $\eta_{i,j}$ is the h^{th} column of LP_L^t and that (θ_p, ι_p) is the ordered pair in S_{next} such that θ_p is the h^{th} column of RP_R (so that $1 \leq p \leq m_{\text{next}}$). A new tuple is now added to S_{lan} by ensuring that vectors are orthogonal to vectors in other new tuples,

$$\begin{aligned} \eta_{i,j} &:= \eta_{i,j} - \sum_{k=1}^{h-1} \eta_{i,j}^t \beta_{m_{\text{old}}+k} \cdot \gamma_{m_{\text{old}}+k}, \\ \theta_p &:= \theta_p - \sum_{k=1}^{h-1} \gamma_{m_{\text{old}}+k}^t \iota_p \cdot \alpha_{m_{\text{old}}+k}, \end{aligned} \quad (17)$$

updating ι_p in the same way (to ensure that $\iota_p = A\theta_p$ once again), and appending triple $(\theta_p, \iota_p, \eta_{i,j}/(\eta_{i,j}^t \iota_p))$ onto S_{lan} .

2.1.4 Updating U and S_{elim} (Steps 9–11)

Let m_{old} and m_{lan} be the lengths of the sequence S_{lan} before and after the previous step, respectively. If all the ordered pairs in S_{next} were matched with vectors using U in step 8, above, then computation continues with an orthogonalization of the unused vectors from U : For each ordered pair (i, j) remaining in U , the vector $\eta_{i,j}$ is updated,

$$\eta_{i,j} := \eta_{i,j} - \sum_{h=m_{\text{old}}+1}^{m_{\text{lan}}} \eta_{i,j}^t \beta_h \cdot \gamma_h \quad (18)$$

in order to ensure that $\eta_{i,j}^t A \alpha_h = 0$ for $1 \leq h \leq m_{\text{lan}}$.

On the other hand, if one or more of the ordered pairs in the sequence S_{next} was not matched, then every unmatched ordered pair (θ, ι) is updated by setting

$$\theta := \theta - \sum_{h=m_{\text{old}}+1}^{m_{\text{lan}}} \gamma_h^t \iota \cdot \alpha_h, \quad \iota := \iota - \sum_{h=m_{\text{old}}+1}^{m_{\text{lan}}} \gamma_h^t \theta \cdot \beta_h \quad (19)$$

to ensure that $\gamma_h^t A \theta = 0$ for $1 \leq h \leq m_{\text{lan}}$ and $\iota = A\theta$ once again. The resulting pair (θ, ι) is then appended to S_{elim} .

2.1.5 Updating S_{next} (Step 12)

To update S_{next} , the pair (θ, ι) is considered where $\theta = \beta_i$ for each triple $(\alpha_i, \beta_i, \gamma_i)$ appended to S_{lan} during the last execution of step 8, and where $\theta = \zeta_i$ for each pair (ϵ_i, ζ_i) appended to S_{elim} if step 11 was executed. The matrix A is applied as an operator to set $\iota = A\theta$ in each case.

Each pair is updated by setting

$$\theta := \theta - \sum_{h=\text{low}_R}^{m_{\text{lan}}} \gamma_h^t \iota \cdot \alpha_h, \quad \iota := \iota - \sum_{h=\text{low}_R}^{m_{\text{lan}}} \gamma_h^t \theta \cdot \beta_h \quad (20)$$

```

13. while ( $S_{\text{next}}$  is not empty) do
14.   if ( $m_{\text{elim}} > \delta$ ) then
15.     Report failure and halt
16.   else
17.      $S_{\text{curr}} := S_{\text{next}}$ ; set  $S_{\text{next}}$  to be empty
18.     for each pair  $(\theta, \iota)$  in  $S_{\text{curr}}$  do
19.       if ( $\iota$  is not a linear
20.         combination of  $\zeta_1, \dots, \zeta_{m_{\text{elim}}}$ ) then
21.         Append  $(\theta, \iota)$  onto  $S_{\text{elim}}$ ;  $m_{\text{elim}}++$ 
22.         Use  $(\iota, A\iota)$  to produce a new value to
           be appended onto  $S_{\text{next}}$ 
23.       else
24.         Use  $(\theta, \iota)$  and  $S_{\text{elim}}$  to extend  $S_{\text{null}}$ 
25.       end if
26.     end for
27.   end if
28. Set  $m_{\text{next}}$  to be the length of the sequence  $S_{\text{next}}$ 
29. end while

```

Figure 2: Elimination Phase of the Main Algorithm

where

$$\text{low}_R = \max(m_{\text{lan}} - 6\ell - r + 1, 1), \quad (21)$$

so that $\gamma_h^t A\theta = 0$ for $1 \leq h \leq m_{\text{lan}}$ and $\iota = A\theta$ once again.

Finally, Gaussian elimination is applied to the second entries of ordered pairs, updating first entries along the way, to look for linear dependencies. Vectors are appended to S_{null} (as appropriate) as entries of the null space are found. As a result of this process S_{next} is as shown in Equation (10), the vector space spanned by the vectors

$$\delta_1, \dots, \delta_{m_{\text{null}}}, \theta_1, \dots, \theta_{m_{\text{elim}}}$$

is unchanged, and $\iota_1, \dots, \iota_{m_{\text{elim}}}$ are linearly independent.

2.2 Details of the Elimination Phase

The major stages of the elimination phase are shown in Figure 2. This consists of a loop that continues as long as the sequence S_{next} is nonempty. Each pair (θ, ι) in S_{next} is appended to the sequence

$$S_{\text{elim}} = (\epsilon_1, \zeta_1), \dots, (\epsilon_{m_{\text{elim}}}, \zeta_{m_{\text{elim}}})$$

if the vectors $\zeta_1, \dots, \zeta_{m_{\text{elim}}}, \iota$ are linearly independent. A new ordered pair (θ', ι') is appended to the sequence S_{next} in this case as well: One initially sets $\theta' = \iota$ and $\iota' = A\theta'$ and then updates these vectors by setting

$$\theta' = \theta' - \sum_{h=\text{low}_R}^{m_{\text{lan}}} \gamma_h^t \iota' \cdot \alpha_h, \quad \iota' = \iota' - \sum_{h=\text{low}_R}^{m_{\text{lan}}} \gamma_h^t \iota' \cdot \beta_h \quad (22)$$

for low_R as at line (21), so that $\gamma_h^t A\theta' = 0$ for $1 \leq h \leq m_{\text{lan}}$.

On the other hand, if $\zeta_1, \dots, \zeta_{m_{\text{elim}}}, \iota$ are linearly dependent, then these values are used to discover a vector $\delta \in \mathbb{F}_q^{n \times 1}$ such that $A\delta = 0$. This is appended to S_{null} if it is not a linear combination of the vectors that have already been included in this sequence.

In order to bound the cost of this phase of the algorithm, the algorithm reports **failure** and terminates if the length of the sequence S_{elim} exceeds δ . Otherwise it terminates, successfully, if S_{next} is the empty sequence at the end of any execution of the body of the loop.

2.3 Analysis of the Algorithm

Suppose the Krylov space $\mathcal{K}$ generated by $v_1, \dots, v_r$ with respect to A has dimension d_{Krylov} and the intersection $\mathcal{N}$ of this space and the null space of A has dimension d_{Null} .

THEOREM 2.1. *Let $A \in \mathbb{F}_q^{n \times n}$ let $v_1, \dots, v_r \in \mathbb{F}_q^{n \times 1}$, and let δ be an integer such that $\delta \geq 2$. If the above algorithm is applied with these values as inputs then either the algorithm generates sequences S_{lan} , S_{null} and S_{elim} satisfying the conditions at lines (1)–(6), above, so that*

$$\alpha_1, \dots, \alpha_{m_{\text{lan}}}, \delta_1, \dots, \delta_{m_{\text{null}}}, \epsilon_1, \dots, \epsilon_{m_{\text{elim}}}$$

*is a basis for the Krylov space $\mathcal{K}$ generated by $v_1, \dots, v_r$ and $\delta_1, \dots, \delta_{m_{\text{null}}}$ is a basis for the intersection of this Krylov space and the null space of A , or the algorithm reports **failure**. The probability of **failure** is at most $7q^{-\delta}$.*

The algorithm uses at most $d_{\text{Krylov}} - d_{\text{Null}} + r$ applications of A , and at most $d_{\text{Krylov}} - d_{\text{Null}} + \ell - \lceil \log_q n \rceil - \delta$ applications of A^t to vectors, where ℓ is the left block size used. It uses $O(n^2(\ell + \delta))$ additional operations in $\mathbb{F}_q$ and requires space to store $O(\ell + \delta)$ vectors in $\mathbb{F}_q^{n \times 1}$ in the worst case.

PROOF (SKETCH). Let $\mathcal{W}_i$ be the vector space spanned by the vectors $A^j v_k$ such that $0 \leq j \leq i$ and $1 \leq k \leq r$. Then one can see by inspection of the algorithm that $\mathcal{W}_i$ is included in the vector space spanned by the set of vectors

$$S_1 = \{\alpha_1, \dots, \alpha_{m_{\text{lan}}}, \delta_1, \dots, \delta_{m_{\text{null}}}, \epsilon_1, \dots, \epsilon_{m_{\text{elim}}}\} \quad (23)$$

while $\mathcal{W}_{i+1}$ is spanned by $S_2 = S_1 \cup \{\theta_1, \dots, \theta_{m_{\text{next}}}\}$, for the vectors shown in Equations (1)–(6) and (10), and for some integer $i \geq 0$. Initially $i = 0$; the value of i is incremented by every execution of the body of each of the loops shown in Figures 1 and 2. Now, if the algorithm terminates without reporting **failure** then it does so because S_{elim} is the empty sequence and $S_1 = S_2$. Thus the vectors in S_1 span a vector space $\mathcal{W}_{i+1}$ that includes $v_1, \dots, v_r$, is closed under multiplication by A (since $\mathcal{W}_{i+1} = \mathcal{W}_{i+2}$ if $S_1 = S_2$), and that is contained in the Krylov space generated by $v_1, \dots, v_r$. The only such space is the Krylov space itself.

A somewhat more involved argument can be used to establish that if $\mathcal{U}_i$ is the vector space spanned by the vectors $(A^t)^j u_k$ for $0 \leq j \leq i$ and $1 \leq k \leq \ell$, then $\mathcal{U}_{s_L}$ is also the vector space that is spanned by the vectors in the set

$$\{\gamma_j \mid 1 \leq j \leq m_{\text{lan}}\} \cup \{\eta_{h,j} \mid (h,j) \in U\}$$

after each execution of these loop bodies as well.

Consider an initialization of a sequence L_{s_L} at step 5. One can prove by consideration of the algorithm that if i is an integer such that $1 \leq i \leq m_{\text{lan}} - r$ then there exist values $c_s \in \mathbb{F}_q$ for $1 \leq s \leq j = \min(m_{\text{lan}}, i + 2r - 1)$ and $d_t \in \mathbb{F}_q$ for $1 \leq t \leq m_{\text{elim}}$ such that

$$A\alpha_i = \sum_{s=1}^j c_s \alpha_s + \sum_{t=1}^{m_{\text{elim}}} d_t \delta_t.$$

A similar result can be established before any update of S_{next} at step 12 or 20: As a result of the greedy choice of vectors in U (described in Subsection 2.1.3) and the test at step 4 in the Lanczos phase, one can show that if i and j are integers such that $0 \leq i \leq s_L - 4$ and $1 \leq j \leq \ell$ then there exist values $c_s \in \mathbb{F}_q$ for $1 \leq s \leq m_{\text{lan}}$ such that

$$A^t \eta_{i,j} = \sum_{s=1}^{m_{\text{lan}}} c_s \gamma_s.$$

Furthermore, if $1 \leq s \leq m_{\text{lan}}$ and $c_s \neq 0$, then $\gamma_s = \eta_{u,v}$ for integers u and v such that $0 \leq u \leq i+3 \leq s_L - 1$ and $1 \leq v \leq \ell$. Now, since $\kappa^t A(A\lambda) = (A^t \kappa)^t A\lambda$ for all vectors $\kappa, \lambda \in \mathbb{F}_q^{n \times 1}$, the above results can be used to establish that the limited orthogonalizations shown at lines (15), (17), (18), (19), (20) and (22) are sufficient to ensure that the conditions at lines (2) and (6) are satisfied at the end of every execution of the bodies of the loops in Figures 1 and 2.

An inspection of the algorithm now suffices to show that if it terminates without **failure** then the conditions in Equations (1)–(6) hold on termination, the vectors in the set S_1 at line (23) form a basis for the Krylov space, and the vectors in S_{null} form a basis for the intersection of the Krylov space and the null space.

The algorithm can be implemented in such a way that the matrix A is only applied to vectors when the sequence S_{next} is updated, at step 12 or step 20, and so that A^t is only applied to vectors when initializing sequences L_{s_L} at step 5. To bound the number of applications of A , one should note that the length of the sequence S_{next} is initially at most r , this does not increase between executions of either loop body, and that the algorithm terminates if S_{next} is empty at the end of any such execution. Each element of S_{next} is used to extend either S_{lan} or S_{elim} whenever the length of S_{next} is not decreased. Consequently the number of applications of A can be bounded by $m_{\text{lan}} + m_{\text{elim}} + r$. The bound given for the number of these applications follows since $d_{\text{Krylov}} = m_{\text{lan}} + m_{\text{elim}} + m_{\text{null}}$ and $d_{\text{Null}} = m_{\text{null}}$ on successful termination of the algorithm. Similarly, one can establish that the number of applications of A^t at step 5 is at most $m_{\text{lan}} - \ell + |U|$ and that $|U| \leq 2\ell - \lceil \log_q n \rceil - \delta$ on termination. The claimed bound on the number of applications of A^t follows because $m_{\text{lan}} \leq d_{\text{Krylov}} - d_{\text{Null}}$. The claimed bound on the number of additional operations over $\mathbb{F}_q$ can be established using standard bounds on the cost of elimination for rectangular systems and the fact that the lengths of sequences S_{null} , S_{next} and S_{elim} never exceed r , r and δ respectively. Finally, the bound on storage space is established by observing that vectors must only be stored as long as they are needed for the limited orthogonalizations described above, or to complete the elimination phase; only $O(\ell + \delta)$ such vectors are ever required.

The use of rectangular blocking (described at line (7)) can be used to establish reliability: Given any choice of $v_1, \dots, v_r$ and $u_1, \dots, u_\ell$, the algorithm only fails at any point using the matrix A if it would also fail using a matrix $\hat{A}$ with at most r invariant factors (polynomials in $\mathbb{F}_q[x]$) different from 1 or x — namely, the matrix $\hat{A}$ whose image is contained in the Krylov space generated by $v_1, \dots, v_r$ such that $\hat{A}\zeta = A\zeta$ for every vector ζ in the Krylov space. Prior results concerning the reliability of black box algorithms, when applied to matrices with small numbers of invariant factors, can now be used to bound the probability of failure of the Lanczos phase by $2q^{-\delta}$ and that of the elimination phase by $5q^{-\delta}$. $\square$

3. A USEFUL MATRIX PROPERTY

Suppose the Jordan normal form of $A \in \mathbb{F}_q^{n \times n}$ includes exactly m nilpotent Jordan blocks (that is, square matrices with ones on the band immediately above the diagonal, and zeroes elsewhere) with order at least two. Then A is k -derogatory, as defined in Section 1, if and only if $k \leq m$.

There exists a nonsingular matrix $X \in \mathbb{F}_q^{n \times n}$ such that

$$A = X \begin{bmatrix} A_{\text{inv}} & & 0 \\ & A_{\text{nil}} & \\ 0 & & A_{\text{zero}} \end{bmatrix} X^{-1} \quad (24)$$

where

- the matrix A_{inv} is nonsingular with order n_{inv} , for some integer n_{inv} such that $0 \leq n_{\text{inv}} \leq n - 2m$, and acts on a vector space $\mathcal{V}_{\text{inv}}$ with basis $Xe_1, \dots, Xe_{n_{\text{inv}}}$, where e_i is the i^{th} standard unit vector for $1 \leq i \leq n$;
- the matrix A_{nil} is block diagonal with diagonal blocks $J_1, J_2, \dots, J_m$, where J_h is a nilpotent Jordan block with order $n_{\text{nil},h} \geq 2$ for $1 \leq h \leq m$, so that A_{nil} has order $n_{\text{nil}} = n_{\text{nil},1} + \dots + n_{\text{nil},m}$; A_{nil} acts on a vector space $\mathcal{V}_{\text{nil}}$ with basis $Xe_{n_{\text{inv}}+1}, \dots, Xe_{n_{\text{inv}}+n_{\text{nil}}}$; and
- the matrix A_{zero} is a zero matrix with order $n_{\text{zero}} = n - n_{\text{inv}} - n_{\text{nil}}$, which acts on a vector space $\mathcal{V}_{\text{zero}}$ with basis $Xe_{n_{\text{inv}}+n_{\text{nil}}+1}, \dots, Xe_n$.

Since the above matrix X is nonsingular, $\mathbb{F}_q^{n \times 1}$ is the direct sum of the vector spaces $\mathcal{V}_{\text{inv}}$, $\mathcal{V}_{\text{nil}}$, and $\mathcal{V}_{\text{zero}}$, and each vector $\chi \in \mathbb{F}_q^{n \times 1}$ can be written uniquely as the sum of vectors $\chi_{\text{inv}} \in \mathcal{V}_{\text{inv}}$, $\chi_{\text{nil}} \in \mathcal{V}_{\text{nil}}$, and $\chi_{\text{zero}} \in \mathcal{V}_{\text{zero}}$. One can see by a consideration of Equation (24) that $\mathcal{V}_{\text{inv}}$, $\mathcal{V}_{\text{nil}}$ and $\mathcal{V}_{\text{zero}}$ are each closed under multiplication by A . As noted above A acts as an invertible operator in $\mathcal{V}_{\text{inv}}$, a nilpotent operator in $\mathcal{V}_{\text{nil}}$, and the zero operator in $\mathcal{V}_{\text{zero}}$.

LEMMA 3.1. *Let $v_i = v_{\text{inv},i} + v_{\text{nil},i} + v_{\text{zero},i} \in \mathbb{F}_q^{n \times 1}$ where $v_{\text{inv},i} \in \mathcal{V}_{\text{inv}}$, $v_{\text{nil},i} \in \mathcal{V}_{\text{nil}}$, and $v_{\text{zero},i} \in \mathcal{V}_{\text{zero}}$ for $1 \leq i \leq r$.*

- The Krylov space $\mathcal{K}$ that is generated by $v_1, \dots, v_r$ includes both the Krylov space $\mathcal{K}_{\text{inv}} \subseteq \mathcal{V}_{\text{inv}}$ generated by $v_{\text{inv},1}, \dots, v_{\text{inv},r}$ and the Krylov space $\mathcal{K}_{\text{nil}} \subseteq \mathcal{V}_{\text{nil}} \oplus \mathcal{V}_{\text{zero}}$ generated by $v_{\text{nil},1} + v_{\text{zero},1}, \dots, v_{\text{nil},r} + v_{\text{zero},r}$.*
- Suppose that $\kappa_1, \dots, \kappa_h$ is a basis for $\mathcal{K}_{\text{inv}}$ and that $\lambda_1, \dots, \lambda_j$ is a basis for $\mathcal{K}_{\text{nil}}$, where $\mathcal{K}_{\text{inv}}$ and $\mathcal{K}_{\text{nil}}$ are as given in part (a), above. Then the vectors*

$$\kappa_1, \dots, \kappa_h, \lambda_1, \dots, \lambda_j$$

form a basis for the above Krylov space $\mathcal{K}$.

Since A_{nil} is block diagonal with m Jordan blocks on its diagonal, $\mathcal{V}_{\text{nil}}$ is the Krylov space generated by m vectors with respect to A . Henceforth let $\omega_1, \dots, \omega_m$ be a set of vectors in $\mathbb{F}_q^{n \times 1}$ such that $\mathcal{V}_{\text{nil}}$ is the Krylov space generated by $\omega_1, \dots, \omega_m$ with respect to A . We will say that a set of vectors $v_1, \dots, v_r$ *captures* A if there exist vectors $\psi_1, \dots, \psi_m \in \mathcal{V}_{\text{zero}}$ such that the Krylov space generated by $v_1, \dots, v_r$ with respect to A includes $\omega_1 + \psi_1, \dots, \omega_m + \psi_m$.

LEMMA 3.2. *Suppose that $r \geq m + \Delta$ for a positive integer Δ , and that vectors $v_1, \dots, v_r$ are chosen uniformly and independently from $\mathbb{F}_q^{n \times 1}$. Then the set $v_1, \dots, v_r$ captures A with probability at least $1 - 2q^{-\Delta}$.*

4. APPLICATIONS

As described in this section a variety of problems can be solved by augmenting or applying the algorithm described in Section 2.

4.1 Solving a Consistent Linear System

System solving is a standard application of Lanczos algorithms. As shown in this section, a common strategy to solve this problem is easily modified to work with the version of a block Lanczos algorithm that has now been presented.

Suppose now that $b \in \mathbb{F}_q^{n \times 1}$ and that we wish to find a vector $x \in \mathbb{F}_q^{n \times 1}$ such that $Ax = b$.

LEMMA 4.1. *Suppose that $v_1 = b$, that $v_2, \dots, v_r$ capture A , and that sequences S_{lan} , S_{null} and S_{elim} are obtained by applying the algorithm from Section 2 on input vectors $v_1, \dots, v_r$, so that they satisfy Equations (1)–(6). Let $\chi_{\text{lan}} = \sum_{i=1}^{m_{\text{lan}}} c_i \alpha_i$, where $c_i = \gamma_i^\dagger b$ for $1 \leq i \leq m_{\text{lan}}$. Then the system $Ax = b$ is consistent if and only there exist elements $d_1, \dots, d_{m_{\text{elim}}}$ of $\mathbb{F}_q$ such that*

$$d_1 \zeta_1 + \dots + d_{m_{\text{elim}}} \zeta_{m_{\text{elim}}} = b - c_1 \beta_1 - \dots - c_{m_{\text{lan}}} \beta_{m_{\text{lan}}}.$$

Furthermore, if $\chi_{\text{elim}} = d_1 \epsilon_1 + \dots + d_{m_{\text{elim}}} \epsilon_{m_{\text{elim}}}$ for values $d_1, \dots, d_{m_{\text{elim}}}$ as above, and $\chi = \chi_{\text{lan}} + \chi_{\text{elim}}$, then $A\chi = b$.

Consider now an algorithm in which one begins by choosing $v_1 = b$ and choosing vectors $v_2, \dots, v_r$ uniformly and independently from $\mathbb{F}_q^{n \times 1}$. Computation continues with an augmented version of the algorithm described in Section 2 in which the vector b is provided as an additional input and two additional vectors, χ_{lan} and ψ_{elim} , are maintained throughout the elimination phase: Initially $\chi_{\text{lan}} = 0$ and $\psi_{\text{elim}} = b$. As each new element $(\alpha_i, \beta_i, \gamma_i)$ is included in the sequence S_{lan} the values χ_{lan} and ψ_{elim} are updated by setting

$$\chi_{\text{lan}} := \chi_{\text{lan}} + c_i \alpha_i \quad \text{and} \quad \psi_{\text{elim}} := \psi_{\text{elim}} - c_i \beta_i,$$

where $c_i = \gamma_i^\dagger b$, in order to ensure that $A\chi_{\text{lan}} + \psi_{\text{elim}} = b$. Now, on termination of the Lanczos phase,

$$\psi_{\text{elim}} = b - c_1 \beta_1 - \dots - c_{m_{\text{lan}}} \beta_{m_{\text{lan}}}.$$

On termination of the elimination phase one should therefore solve a system of linear equations to try to find values $d_1, \dots, d_{m_{\text{elim}}}$ such that

$$d_1 \zeta_1 + \dots + d_{m_{\text{elim}}} \zeta_{m_{\text{elim}}} = \psi_{\text{elim}};$$

one can set $\chi_{\text{elim}} := d_1 \epsilon_1 + \dots + d_{m_{\text{elim}}} \epsilon_{m_{\text{elim}}}$ and return $\chi = \chi_{\text{lan}} + \chi_{\text{elim}}$ as a solution for the given system if these values are found, returning **failure** otherwise.

Suppose that $r \geq m + \Delta + 1$ for a positive integer Δ ; then Lemma 3.2 implies that $v_2, \dots, v_r$ capture A with probability at least $1 - 2q^{-\Delta}$. If this is the case then it follows by Lemma 4.1 that a vector χ is returned by the above algorithm such that $A\chi = b$ whenever the system $Ax = b$ is consistent, with **failure** reported if the system is inconsistent. The cost of the augmented algorithm is not significantly different from that of the original version so that, apart from the need to choose an additional $r - 1$ vectors uniformly and independently from $\mathbb{F}_q^{n \times 1}$, it is as described in Section 2.

4.2 Bounding the Number of Nilpotent Blocks

Next consider the problem of bounding the number of nontrivial nilpotent blocks in the Jordan normal form of A , or computing this value exactly if it is small.

LEMMA 4.2. *Suppose $v_1, \dots, v_r$ capture A ; then the intersection of the Krylov space generated by $Av_1, \dots, Av_r$ and the null space of A is equal to the intersection $\mathcal{Z}$ of the image of A and the null space of A . The dimension of this space is equal to the number of nilpotent blocks with order at least two in the Jordan normal form of A .*

Once again, consider an augmented version of the algorithm in Section 2. This should begin by selecting r vectors $w_1, \dots, w_r$ uniformly and independently from $\mathbb{F}_q^{n \times 1}$, and by setting v_i to be Aw_i for $1 \leq i \leq r$. The sequences S_{lan} , S_{null} , S_{elim} , and S_{next} described in Section 2 should be re-defined so that they include “preimages” of some of the vectors currently included. In particular, each tuple $(\alpha_i, \beta_i, \gamma_i)$ of S_{lan} should be replaced by a tuple $(\alpha_{\text{pre},i}, \alpha_i, \beta_i, \gamma_i)$ such that $A\alpha_{\text{pre},i} = \alpha_i$. Similarly each vector δ_i in S_{null} should be replaced by an ordered pair $(\delta_{\text{pre},i}, \delta_i)$, each pair (ϵ_i, ζ_i) in S_{elim} should be replaced by a tuple $(\epsilon_{\text{pre},i}, \epsilon_i, \zeta_i)$, and each pair (θ_i, ι_i) in S_{next} should be replaced by a tuple $(\theta_{\text{pre},i}, \theta_i, \iota_i)$, where $A\delta_{\text{pre},i} = \delta_i$, $A\epsilon_{\text{pre},i} = \epsilon_i$, and $A\theta_{\text{pre},i} = \theta_i$ respectively. On termination the sequence S_{null} will consist of a sequence of m_{null} ordered pairs $(\delta_{\text{pre},1}, \delta_1), \dots, (\delta_{\text{pre},m_{\text{null}}}, \delta_{m_{\text{null}}})$ such that $A\delta_{\text{pre},i} = \delta_i$ and $\delta_i = 0$ for $1 \leq i \leq m_{\text{null}}$ and such that $\delta_1, \dots, \delta_{m_{\text{null}}}$ are linearly independent — establishing that the vector space $\mathcal{Z}$ mentioned in Lemma 4.2 has dimension at least m_{null} , so that A is m_{null} -derogatory.

Once again, Lemma 3.2 can be used to assess the reliability of this algorithm when the number m of nilpotent blocks with order at least two in the Jordan form is small: If $r \geq m + \Delta$ for a positive integer Δ then $m_{\text{null}} = m$ with probability at least $1 - 2q^{-\Delta}$. The next lemma is of use when m is large.

LEMMA 4.3. *Suppose that the above algorithm is applied when $m \geq r + \Delta$ for a positive integer Δ . Then $m_{\text{null}} \leq r$, and $m_{\text{null}} < r$ with probability at most $2q^{-\Delta}$.*

Hence one can be reasonably confident that $m_{\text{null}} = m$ if m_{null} is significantly smaller than r on termination; with high probability m_{null} will be equal to r if r is significantly smaller than m . If $r - m_{\text{null}}$ is small but positive on termination then a second execution of the algorithm with a slightly larger number r of vectors should determine the number m of nilpotent blocks.

If the augmented algorithm is implemented with care then only r additional applications of the matrix A as an operator should be required. Apart from this, the cost to execute this algorithm will be as described in Section 2 once again.

4.3 Sampling from the Null Space

Suppose again that $A \in \mathbb{F}_q^{n \times n}$ is a singular matrix with exactly m nontrivial nilpotent blocks in its Jordan normal form and, indeed, that the value of m is known. As usual it will be assumed that one can sample elements uniformly and independently from $\mathbb{F}_q$.

Suppose as well that we are given an integer $d > 0$ and that we wish to generate a sequence $\psi_1, \dots, \psi_d$ such that $A\psi_i = 0$ for $1 \leq i \leq d$, and such that the conditional probability that every such sequence is generated (given that the algorithm does not report **failure**, instead) is q^{-kd} , where k is the (generally unknown) dimension of the null space. An algorithm to compute these values will begin by selecting vectors $v_1, \dots, v_{m+d}$ uniformly and independently from $\mathbb{F}_q^{n \times 1}$, and carrying out the following three steps.

1. Confirm that $v_1, \dots, v_{m+d}$ capture A and generate a basis $\varphi_1, \dots, \varphi_m$ for the intersection $\mathcal{Z}$ of the image and null space of A . Report **failure** and terminate, instead, if $v_1, \dots, v_{m+d}$ do not capture A .
2. Generate vectors $\psi'_1, \dots, \psi'_d$ such $\psi'_i = \psi'_{i,\text{nil}} + \psi'_{i,\text{zero}}$ where $\psi'_{i,\text{nil}} \in \mathcal{V}_{\text{nil}}$ and $\psi'_{i,\text{zero}} \in \mathcal{V}_{\text{zero}}$ for $1 \leq i \leq d$,

and the conditional probability that $\psi_{1,\text{zero}}, \dots, \psi_{d,\text{zero}}$ is generated (given that **failure** is not reported) is $q^{-(k-m)d}$ for every such sequence of d vectors in $\mathcal{V}_{\text{zero}}$.

3. Use the above to generate the sequence $\psi_1, \dots, \psi_d$ to be returned.

Each of these steps is described in more detail below.

Step 1. The algorithm of Section 4.2 can be used here with vectors $v_1, \dots, v_{m+d}$ as input. If $m_{\text{null}} \neq m$ on termination then **failure** should be reported. Otherwise one can set $\varphi_i = \delta_i$ for $1 \leq i \leq m$, for the vectors $\delta_1, \dots, \delta_m$ included in the entries of the sequence S_{null} , to provide a basis for $\mathcal{Z}$.

Step 2. Notice that if the vectors $v_1, \dots, v_{m+d}$ capture A then $\mathcal{Z}$ is a subset of the intersection $\mathcal{N}$ of the Krylov space generated by $v_1, \dots, v_r$ and the null space of A , so that $\mathcal{N}$ has a basis

$$\varphi_1, \dots, \varphi_m, \tau_1, \dots, \tau_e$$

where $\tau_j = \tau_{j,\text{nil}} + \tau_{j,\text{zero}}$ for $\tau_{j,\text{nil}} \in \mathcal{Z}$ and $\tau_{j,\text{zero}} \in \mathcal{V}_{\text{zero}}$ for $1 \leq j \leq e$, and for $e \leq d$. Indeed, such a basis can be obtained (on completion of Step 1) from an arbitrary basis for $\mathcal{N}$ by using it to extend the basis $\varphi_1, \dots, \varphi_m$ for $\mathcal{Z}$. The algorithm for step 2 will begin by generating vectors $\tau_1, \dots, \tau_e$ as above by applying the algorithm in Section 2 with $v_1, \dots, v_{m+d}$ to obtain a basis for $\mathcal{N}$ and then using this basis to carry out the process sketched above.

Let $L_{\text{zero}} \in \mathbb{F}_q^{n \times e}$ be a matrix with the above vectors $\tau_1, \dots, \tau_e$ as columns, and let $R_{\text{zero}} \in \mathbb{F}_q^{e \times d}$ be a matrix that is selected uniformly (and independently) from the set of all $e \times d$ matrices with full rank e . The columns of the product $L_{\text{zero}} R_{\text{zero}} \in \mathbb{F}_q^{n \times d}$ are used as the vectors $\psi'_1, \dots, \psi'_d$ to be returned as the results of this step.

In order to assess the reliability of this step, consider a subspace $\mathcal{W}$ of $\mathcal{V}_{\text{zero}}$ with dimension at most d . Let $P_1(\mathcal{W})$ be the conditional probability that $\mathcal{W}$ is the vector space with basis $\tau_{1,\text{zero}}, \dots, \tau_{e,\text{zero}}$ when the above process is applied (given that the process does not report **failure**), and let $P_2(\mathcal{W})$ be the probability that $\mathcal{W}$ is the space spanned by a sequence of vectors $\sigma_1, \dots, \sigma_d$ that are generated uniformly and independently from $\mathcal{V}_{\text{zero}}$.

LEMMA 4.4. $P_1(\mathcal{W}) = P_2(\mathcal{W})$ for every vector space $\mathcal{W} \subseteq \mathcal{V}_{\text{zero}}$ with dimension at most d .

Having selected a subspace $\mathcal{W}$ of $\mathcal{V}_{\text{zero}}$ and a matrix L_{zero} with columns $\tau_1, \dots, \tau_e$, where $\tau_i = \tau_{i,\text{nil}} + \tau_{i,\text{zero}}$ with $\tau_{i,\text{nil}} \in \mathcal{Z}$ for $1 \leq i \leq e$, and where $\tau_{1,\text{zero}}, \dots, \tau_{e,\text{zero}}$ is a basis for $\mathcal{W}$, it now suffices to note that we should choose each sequence of vectors $\psi'_1, \dots, \psi'_d$ of length d that spans the column space of L_{nil} with the same probability, in order to satisfy the requirements of step 2. Every such sequence forms the columns of a matrix $L_{\text{zero}} R_{\text{zero}}$ for exactly one matrix $R_{\text{zero}} \in \mathbb{F}_q^{e \times d}$ with maximal rank e , so the completion of the process sketched above (and, in particular, the use of the above matrix R_{zero}) results in vectors $\psi'_1, \dots, \psi'_d$ such that $\psi'_i = \psi'_{i,\text{nil}} + \psi_{i,\text{zero}}$ where $\psi'_{i,\text{nil}} \in \mathcal{V}_{\text{nil}}$ and $\psi_{i,\text{zero}} \in \mathcal{V}_{\text{zero}}$ for $1 \leq i \leq d$, with each sequence $\psi_{1,\text{zero}}, \dots, \psi_{d,\text{zero}}$ of vectors in $\mathcal{V}_{\text{zero}}$ produced with probability $q^{-(k-m)d}$, as required.

Step 3. Let $L_{\text{nil}} \in \mathbb{F}_q^{n \times m}$ be a matrix whose columns are the vectors $\varphi_1, \dots, \varphi_m$ in the basis for $\mathcal{Z}$ produced in Step 1, and let $R_{\text{nil}} \in \mathbb{F}_q^{m \times d}$ be a matrix whose entries are chosen uniformly and independently from $\mathbb{F}_q$. The values $\psi_1, \dots, \psi_d$ should be returned where $\psi_i = \psi'_i + \psi''_i$ for $1 \leq i \leq d$,

$\psi'_1, \dots, \psi'_d$ are the vectors that were generated using step 2, above, and where $\psi''_1, \dots, \psi''_d$ are the columns of the product $L_{\text{nil}} R_{\text{nil}} \in \mathbb{F}_q^{n \times d}$.

To see that this is correct, note that $\psi_i = \psi_{i,\text{nil}} + \psi_{i,\text{zero}}$ where $\psi_{i,\text{nil}} = \psi'_{i,\text{nil}} + \psi''_i$ and where $\psi_{i,\text{zero}}$ is as described in the discussion of Step 2, above, for $1 \leq i \leq d$. The selection of R_{nil} above ensures that the vectors $\psi_{1,\text{nil}}, \dots, \psi_{d,\text{nil}}$ are selected uniformly from $\mathcal{Z}$ and, furthermore, that they are selected independently of each other and of the prior selection of $\psi_{1,\text{zero}}, \dots, \psi_{d,\text{zero}}$ — so that each sequence $\psi_1, \dots, \psi_d$ of vectors in $\mathcal{N}$ is selected with probability q^{-kd} , as desired.

Apart from the applications of the algorithms described in Sections 4.2 and 2 (in steps 1 and 2, respectively), the computationally most expensive step in this process is the generation of the matrix R_{zero} described in step 2. To ensure that this matrix has full rank it is sufficient to choose rows uniformly and randomly, rejecting a row and trying again whenever the new row is a linear combination of the rows that have already been accepted. The likelihood that a row is rejected is at most q^{-1} so it can be established that the expected number of values that must be selected uniformly and independently from $\mathbb{F}_q$ to compute R_{zero} is in $O(de)$ and the expected number of additional operations in $\mathbb{F}_q$ needed is in $O(d^3)$. If $d\delta \in O(n)$, where δ is the positive integer input for the main algorithm described in Section 2, then one can produce a version of the algorithm whose worst-case cost is asymptotically no worse than the original version of the algorithm, while increasing the probability of **failure** by at most $q^{-\delta}$, by requiring that the algorithm fails if 2δ attempts to generate any row of the matrix all fail.

Finally, it should be noted that the probability of **failure** of the above algorithm depends on the input parameter d , since the likelihood that $v_1, \dots, v_{m+d}$ do not capture A has only been bounded by $2q^{-d}$. If one wishes to establish an error bound in $O(q^{-\delta})$ for a larger integer δ then it suffices to replace d with δ in the above construction and then ignore the final $\delta - d$ vectors in $\mathcal{N}$ that are generated.

4.4 Certifying Inconsistency

Consider next the problem of certifying that a given system $Ax = b$ is *not* consistent, that is, that b does not belong to the column space of A .

If the matrix A is m -nonderogatory then so is its transpose and, as observed by Giesbrecht, Lobo and Saunders [6], the probability that $\mu^t b \neq 0$ is at least $1 - 1/q$ if the system of linear equations $Ax = b$ is inconsistent and μ is a uniformly and randomly chosen element of the null space of A^t . No such vector μ can exist if the system is consistent.

An algorithm that certifies inconsistency of a system $Ax = b$ for an m -nonderogatory matrix A can now be obtained using the algorithm given in Section 4.3, above: This should be used to produce a sequence of Δ vectors $\psi_1, \psi_2, \dots, \psi_\Delta$ that are uniformly and independently selected from the null space of A^t . If the number of nontrivial nilpotent blocks in A supplied to the algorithm was correct and the algorithm did not report **failure**, then a vector ψ_i such that $\psi_i^t A = 0$ but $\psi_i^t b \neq 0$ will have been produced, for some integer i between 1 and Δ , with probability at least $1 - q^{-\Delta}$.

As described here the algorithm can fail for a variety of reasons. This can be addressed by combining it with an attempt to solve the given system using the algorithm in Subsection 4.1. The result is an algorithm — whose inputs should include the matrix A , vector b , and a positive inte-

ger m — and which produces either

- (i) a vector x such that $Ax = b$,
- (ii) a vector μ such that $\mu^t A = 0$ but $\mu^t b \neq 0$,
- (iii) a proof that the matrix A is m -derogatory, as described above, or
- (iv) a report of **failure**,

and where the only reason for **failure** is an unlucky choice of the random values that have been selected. Consequently repeated trials of this will eventually result in one of outputs (i), (ii), or (iii). In the event of (iii) a user should presumably try again with a larger value of m or apply a preconditioner in order to bring A into a more manageable form.

5. FURTHER WORK

An implementation of this algorithm using LinBox is being developed and will be available at the author's web site.

Improvements of the algorithm in Section 2 would certainly be desirable. For example, while it is *sufficient* to store approximately 7ℓ pairs of vectors to be used for the limited orthogonalizations described in Section 2, I do not know whether it is necessary and, of course, the number of vectors to be stored should be reduced if possible. It is also possible that the elimination phase of the algorithm in Section 2 can be eliminated — the limited testing of the implementation now being developed has suggested that it is not needed in practice.

The algorithms of Coppersmith [3] or Montgomery [8] suggest that a block Lanczos algorithm for symmetric matrices is (at least potentially) much simpler than the algorithm that has been presented here. Is there an alternative (randomized) process that can be used to symmetrize the input, resulting in a block Lanczos algorithm more closely resembling that of either Coppersmith or Montgomery, that is also reliable for computations over small fields?

All that noted, it is not clear that a block Lanczos algorithm is necessary to establish the results that have been described: It is certainly plausible (but, to my knowledge, not yet verified) that an existing block Wiedemann algorithm could also be used to carry out the computations described in Section 4.

The small field preconditioner mentioned at the beginning of this paper is arguably more expensive than is desirable, but it also achieves a stronger matrix property than is used here, in that it ensures, with high probability, that the number of nontrivial invariant factors of the preconditioned matrix is small. Are there other, less expensive small field preconditioners, that are sufficient to ensure the weaker condition that the preconditioned matrix is nonderogatory?

A final theoretical question concerns a property of the null space of a matrix: Is it possible to discover the dimension of the null space (and, therefore, the rank of the given matrix)

using only the fact that the matrix is nonderogatory, and without preconditioning? Note that the algorithm to sample from the null space in Section 4 does not require this value and, as far as I can tell, fails to provide any information that could be used to discover this in general. I suspect that the answer to this question is “no,” but have no idea of how to prove this.

6. REFERENCES

- [1] J. P. Buhler, J. H. W. Lenstra, and C. Pomerance. Factoring integers with the number field sieve. In *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Computer Science*, pages 50–94. Springer-Verlag, 1993.
- [2] L. Chen, W. Eberly, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. Efficient matrix preconditioners for black box linear algebra. *Linear Algebra and Its Applications*, 343–344:119–146, 2002.
- [3] D. Coppersmith. Solving linear equations over $\text{GF}(2)$: Block Lanczos algorithm. *Linear Algebra and Its Applications*, 192:33–60, 1993.
- [4] W. Eberly. Reliable Krylov-based algorithms for matrix null space and rank. In *Proceedings, 2004 International Symposium on Symbolic and Algebraic Computation*, pages 127–134, Santander, Spain, 2004.
- [5] W. Eberly. Yet another block Lanczos algorithm: How to simplify the computation and reduce reliance on preconditioners in the small field case. Technical Report 2010-957-06, Department of Computer Science, University of Calgary, 2010. Available online at www.cpsc.ucalgary.ca/~eberly/Research/publications.php.
- [6] M. Giesbrecht, A. Lobo, and B. D. Saunders. Certifying inconsistency of sparse linear systems. In *Proceedings, 1998 International Symposium on Symbolic and Algebraic Computation*, pages 113–119, University of Rostock, Germany, 1998.
- [7] B. Hovinen and W. Eberly. A reliable block Lanczos algorithm over small finite fields. In *Proceedings, 2005 International Symposium on Symbolic and Algebraic Computation*, pages 177–184, Beijing, China, 2005.
- [8] P. L. Montgomery. A block Lanczos algorithm for finding dependencies over $\text{GF}(2)$. In *Advances in Cryptology—EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 106–120. Springer-Verlag, 1995.
- [9] G. Villard. Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems. In *Proceedings, 1997 International Symposium on Symbolic and Algebraic Computation*, pages 32–39, Maui, Hawaii, 1997.
- [10] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, IT-32:54–62, 1986.

Liouvillian Solutions of Irreducible Second Order Linear Difference Equations

Mark van Hoeij*
Department of Mathematics
Florida State University
Tallahassee, FL 32306, USA
hoeij@math.fsu.edu

Giles Levy
Department of Mathematics
Florida State University
Tallahassee, FL 32306, USA
glevy@math.fsu.edu

ABSTRACT

In this paper we give a new algorithm to compute Liouvillian solutions of linear difference equations. The first algorithm for this was given by Hendriks in 1998, and Hendriks and Singer in 1999. Several improvements have been published, including a paper by Cha and van Hoeij that reduces the combinatorial problem. But the number of combinations still depended exponentially on the number of singularities. For irreducible second order equations, we give a short and very efficient algorithm; the number of combinations is 1.

Categories and Subject Descriptors

G.2.1 [Combinatorics]: [Recurrences and difference equations];
I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—Algebraic algorithms

General Terms

Algorithms

1. INTRODUCTION

Let τ denote the shift operator: $\tau(f(n)) = f(n+1)$. An operator $L_1 = \tau^2 + a_1\tau + a_0$ with $a_0, a_1 \in \mathbb{C}(n)$ corresponds to a recurrence relation $u(n+2) + a_1(n)u(n+1) + a_0(n)u(n) = 0$. Algorithms for finding rational resp. hypergeometric resp. Liouvillian solutions have been given in [2, 4] resp. [7, 5, 6] resp. [11, 12, 3, 14]. The algorithms for hypergeometric solutions use a combinatorial search, where each of the combinations involves computing polynomial solutions. The algorithms for Liouvillian solutions are also combinatorial in nature, either because they call an algorithm for hypergeometric solutions [11, 12, 3], or perform a reduced (but still exponential) combinatorial search [14].

A second order operator L_1 is irreducible if and only if it does not have hypergeometric solutions. If L_1 is irreducible, then the task of finding a gauge transformation (see Definition 8) from L_1 to an operator of the form $L_2 = \tau^2 + b_0$ with $b_0 \in \mathbb{C}(n)$ is equivalent to finding Liouvillian solutions of L_1 ([12, Lemma 4.1]). Finding

*Supported by NSF grant 0728853

such a transformation is useful because $\tau^2 + b_0$ is easily solved with interlaced hypergeometric terms as follows:

Let the factorization of b_0 into monic linear factors be $c \frac{p_1 p_2 \dots p_i}{q_1 q_2 \dots q_j}$, $c \in \mathbb{C}$, then $\tau^2 + b_0$ has solutions

$$\frac{\Gamma(p_1/2)\Gamma(p_2/2)\dots\Gamma(p_i/2)}{\Gamma(q_1/2)\Gamma(q_2/2)\dots\Gamma(q_j/2)} \cdot c^{n/2} \cdot \begin{cases} k_1, & \text{if } n \text{ even} \\ k_2, & \text{if } n \text{ odd} \end{cases} \quad (1)$$

where k_1, k_2 are arbitrary constants.

As an example we look at A099364 from ‘The On-Line Encyclopedia of Integer Sequences’ or OEIS ([1]): A099364 is a sequence named “An inverse Chebyshev transform of $(1-x)^2$ ” and satisfies $(n+6)u(n+2) + 2u(n+1) - (8+4n)u(n) = 0$. Our implementation ([16] or [15]) returns the solution:

$$u(n) = \left(\frac{1}{6}n + \frac{5}{6}\right)v(n) - \left(\frac{1}{12}n + \frac{1}{2}\right)v(n+1),$$

$$\text{where } v(n+2) - \frac{4(n+2)}{n+7}v(n) = 0 \text{ and } \{v(0) = 0, v(1) = -2\}$$

so $v(n)$ can be represented as in equation (1). Prior algorithms for Liouvillian solutions will also solve this equation, using a search that depends exponentially on the number of finite singularities (see [14] for more details). This paper introduces a new method for finding Liouvillian solutions. It works for irreducible linear difference equations of order 2. Our algorithm is short and easy to implement [16], but its primary benefit is that it is very efficient because Corollary 1 allows us to reduce the number of combinations to 1.

The idea behind the algorithm is as follows (for notations see Section 2). Suppose that $L_1 = \tau^2 + a_1\tau + a_0$ and $L_2 = \tau^2 + b_0$ with $b_0, a_0, a_1 \neq 0$, and b_0 unknown. If there exists a gauge transformation $G: L_1 \rightarrow L_2$ then there is an induced transformation between their symmetric squares $L_1^{\otimes 2}, L_2^{\otimes 2}$, which are of order 3 and 2 respectively. The induced mapping must therefore have a one dimensional kernel, which corresponds to a hypergeometric solution of $L_1^{\otimes 2}$. But we can find this hypergeometric solution without a combinatorial search, because Corollary 1 reduces the problem to computing a rational solution.

2. PRELIMINARIES

More detailed preliminaries can be found in [15].

DEFINITION 1. τ will refer to the shift operator acting on $\mathbb{C}(n)$ and $\text{Mat}_{a \times b}(\mathbb{C}(n))$ by $\tau: n \mapsto n+1$.

An operator $L = \sum_i a_i \tau^i$ acts as $Lu(n) = \sum_i a_i u(n+i)$.

DEFINITION 2. $\mathbb{C}(n)[\tau]$ is the ring of linear difference operators where ring multiplication is composition of operators $L_1 L_2 = L_1 \circ L_2$, e.g. $(\tau - a(n))(\tau - b(n)) = \tau^2 - (a(n) + b(n+1))\tau + a(n)b(n)$.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

We let τ operate on $u(n) \in \mathbb{C}^{\mathbb{N}}$ by $u(n) \mapsto u(n+1)$.

DEFINITION 3. [*'Galois Theory of Difference Equations' Example 1.3 [10]*] Let $S = \mathbb{C}^{\mathbb{N}}/\sim$ where $s_1 \sim s_2$ if there exists $N \in \mathbb{N}$ such that, for all $n > N$, $s_1(n) = s_2(n)$.

The reason for using S is that the dimension of the solution space will be equal to the order of the difference operator (see Theorem 1 below). Working in S also enables us to work in $\mathbb{C}[n][\tau]$ as well as in $\mathbb{C}(n)[\tau]$. In particular, if $L \in \mathbb{C}(n)[\tau]$ and we multiply away the denominators of the coefficients to obtain an element of $\mathbb{C}[n][\tau]$ then the solution space does not change when working in S .

DEFINITION 4. $V(L)$ refers to the solution space of the operator L , i.e. $V(L) = \{u \in S \mid Lu = 0\}$, where S is as in Definition 3.

EXAMPLE 1. For $L = \tau + n + 1$ we write $V(L) = \mathbb{C} \cdot (-1)^n \Gamma(n+1)$ or $V(L) = \mathbb{C} \cdot [1, -1, 2, -6, 24, -120, \dots]$.

DEFINITION 5. A unit is a sequence in S that is invertible, i.e. a sequence that only has finitely many zeros.

THEOREM 1. [*'A=B' Theorem 8.2.1 [9]*] Let $L = \sum_{k=0}^r a_k \tau^k$ be a linear difference operator of order r on S . If a_r and a_0 are units, then $\dim(\ker(L)) = r$.

We can view $\mathbb{C}(n)$ as a subset of S so the theorem applies to $L \in \mathbb{C}(n)[\tau]$ with $a_0, a_r \neq 0$.

DEFINITION 6. A function or sequence $v(n)$ for which $v(n+1)/v(n)$ is a rational function of n will be called a hypergeometric term. Such a $v(n)$ will be called a hypergeometric solution of any $L \in \mathbb{C}(n)[\tau]$ for which $Lv = 0$. It corresponds to a first order right hand factor of L , namely $\tau - r(n)$ where $r(n) = v(n+1)/v(n)$.

A hypergeometric function is a function $\sum_{n=0}^{\infty} v(n)x^n$ where $v(n)$ is a hypergeometric term.

2.1 Gauge Transformations

Let $D = \mathbb{C}(n)[\tau]$. If $L \in D \setminus \{0\}$ then D/DL is a D -module.

DEFINITION 7. L_1 is gauge equivalent to L_2 when D/DL_1 and D/DL_2 are isomorphic as D -modules.

LEMMA 1. L_1 is gauge equivalent to L_2 if and only if there exists $G \in D$ such that $G(V(L_1)) = V(L_2)$ and L_1, L_2 have the same order. Thus G defines a bijection $V(L_1) \rightarrow V(L_2)$.

Note: If $D/DL_1 \cong D/DL_2$ then G in the Lemma corresponds to the image in D/DL_1 of the element $1 \in D/DL_2$.

DEFINITION 8. The bijection defined by G in Lemma 1 above will be called a gauge transformation.

DEFINITION 9. Let $r(n) = cp_1(n)^{e_1} \cdots p_j(n)^{e_j} \in \mathbb{C}(n)$ with $C \in \mathbb{C}$. Let the $e_i \in \mathbb{Z}$, let the $p_i(n)$ be irreducible in $\mathbb{C}[n]$, and let $s_i \in \mathbb{C}$ be the sum of the roots of $p_i(n)$. Then $r(n)$ is said to be in shift normal form if $-\deg(p_i(n)) < \operatorname{Re}(s_i) \leq 0$, for $i = 1, \dots, j$. We denote $\operatorname{SNF}(r(n))$ as the shift normalized form of $r(n)$ which is obtained by replacing each $p_i(n)$ by $p_i(n + k_i)$ for some $k_i \in \mathbb{Z}$ such that $p_i(n + k_i)$ is in shift normal form. Two rational functions, $r_1(n), r_2(n)$ will be called shift equivalent, denoted $r_1(n) \stackrel{\text{SE}}{=} r_2(n)$, if $\operatorname{SNF}(r_1(n)) = \operatorname{SNF}(r_2(n))$.

REMARK 1. Let $r_1, r_2 \in \mathbb{C}(n)$. If $r_1(n) \stackrel{\text{SE}}{=} r_2(n)$ then there exists $r \in \mathbb{C}(n)$ such that $r_1/r_2 = r(n+1)/r(n)$. This is easy to prove if r_1, r_2 are irreducible polynomials. The general case reduces to the irreducible case because the relation $\stackrel{\text{SE}}{=}$ is closed under multiplication, and the same is true for the group of all $r(n+1)/r(n)$.

DEFINITION 10. The companion matrix of a monic difference operator

$$L = \tau^k + a_{k-1}\tau^{k-1} + \cdots + a_0, \quad a_i \in \mathbb{C}(n)$$

which is satisfied by $u(n)$ will refer to the matrix:

$$M = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -a_0 & -a_1 & \cdots & -a_{k-2} & -a_{k-1} \end{pmatrix}. \quad (2)$$

The equation $Lu = 0$ is equivalent to the system $\tau(Y) = MY$ where

$$Y = \begin{pmatrix} u(n) \\ \vdots \\ u(n+k-1) \end{pmatrix}. \quad (3)$$

DEFINITION 11. Let $L = a_k \tau^k + a_{k-1} \tau^{k-1} + \cdots + a_0$, $a_i \in \mathbb{C}(n)$. The determinant of L , $\det(L) := (-1)^k a_0/a_k$, i.e. the determinant of its companion matrix.

LEMMA 2. If there exists a gauge transformation $G: V(L_1) \rightarrow V(L_2)$ then $\det(L_1) \stackrel{\text{SE}}{=} \det(L_2)$.

The proof follows by examining the matrix representations of L_1, L_2 , and G . If $\hat{G}$ is the matrix representation of G then $\det(L_2) = \det(L_1) \tau(\det(\hat{G}))/\det(\hat{G})$, for details see [15].

2.2 Liouvillian Solutions and Symmetric Products

DEFINITION 12 (DEFINITION 6 IN [6]). Let $L_1, L_2 \in \mathbb{C}(n)[\tau]$. The symmetric product (called term product in [15]) $L_1 \otimes L_2$ of L_1 and L_2 is defined as the monic operator $L \in \mathbb{C}(n)[\tau]$ of smallest order such that $L(u_1 u_2) = 0$ for all $u_1, u_2 \in S$ with $L_1 u_1 = 0$ and $L_2 u_2 = 0$.

The following is an example of the symmetric product of a second order $L_1 = \tau^2 + b(n)\tau + c(n)$ with a first order $L_2 = \tau - a(n)$:

$$L_1 \otimes L_2 = \tau^2 + b(n)a(n+1)\tau + c(n)a(n)a(n+1).$$

DEFINITION 13. The symmetric square of L , denoted $L^{\otimes 2}$, will refer to the symmetric product of L and L .

Liouvillian solutions are defined in [12] Section 3.2. For irreducible operators they are characterized by the following theorem.

THEOREM 2. (PROPOSITIONS 31-32 IN [8], OR LEMMA 4.1 IN [12]): An irreducible k 'th order operator L has Liouvillian solutions if and only if its companion matrix is gauge equivalent to one that can be written as

$$M = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -a & 0 & \cdots & 0 & 0 \end{pmatrix}, \quad a \in \mathbb{C}(n).$$

In other words, L is gauge equivalent to $\tau^k + a$.

LEMMA 3. Let $L = a_2\tau^2 + a_1\tau + a_0$, $a_i \in \mathbb{C}[n]$, $a_0, a_2 \neq 0$.

1. If $a_1 \neq 0$ then

$$L^{\otimes 2} = c_3\tau^3 + c_2\tau^2 + c_1\tau + c_0,$$

where:

$$\begin{aligned} c_3 &= a_1(n)a_2(n+1)^2a_2(n) \\ c_2 &= a_1(n+1)a_2(n)(-a_1(n+1)a_1(n) + a_0(n+1)a_2(n)) \\ c_1 &= a_0(n+1)a_1(n)(a_1(n+1)a_1(n) - a_0(n+1)a_2(n)) \\ c_0 &= -a_1(n+1)a_0(n+1)a_0(n)^2. \end{aligned} \quad (4)$$

If $a_1 = 0$ then

$$L^{\otimes 2} = a_2^2\tau^2 - a_0^2. \quad (5)$$

2. $L^{\otimes 2}$ has order: $\begin{cases} 2, & \text{if } a_1 = 0 \\ 3, & \text{if } a_1 \neq 0 \end{cases}$

For a proof of the lemma, which relies on the linear independence of the solutions of L , see [15].

REMARK 1. The proof of the Lemma below illustrates computations in Step 4 of Algorithm FindLiouvillian in Section 3.

LEMMA 4. Let $a \neq 0$. Given a gauge transformation from $L = \tau^2 + a(n)\tau + b(n)$ to $\hat{L} = \tau^2 + r(n)$ one can compute a difference operator mapping $V(L^{\otimes 2})$ onto $V(\hat{L}^{\otimes 2})$.

PROOF. Let $u(n) \in V(L)$ and $v(n) \in V(\hat{L})$ with $v(n) = g_0(n)u(n) + g_1(n)u(n+1)$, then

$$v(n)^2 = g_0(n)^2u(n)^2 + 2g_0(n)g_1(n)u(n)u(n+1) + g_1(n)^2u(n+1)^2.$$

The substitution (obtained by squaring $u(n+2) = -a(n)u(n+1) - b(n)u(n)$):

$$u(n)u(n+1) = \frac{u(n+2)^2 - a(n)^2u(n+1)^2 - b(n)^2u(n)^2}{2a(n)b(n)}$$

yields:

$$\begin{aligned} v(n)^2 &= \frac{g_0(n)(-g_1(n)b(n) + g_0(n)a(n))}{a(n)}u(n)^2 \\ &- \frac{g_1(n)(-g_1(n)b(n) + g_0(n)a(n))}{b(n)}u(n+1)^2 + \frac{g_0(n)g_1(n)}{a(n)b(n)}u(n+2)^2. \end{aligned} \quad (6)$$

Since $V(L^{\otimes 2})$ is spanned by squares, by linear extension equation (6) will define a map from $V(L^{\otimes 2})$ to $V(\hat{L}^{\otimes 2})$. To prove that this map is onto, choose linearly independent $u_1, u_2 \in V(L)$. Applying the gauge transformation $g_0(n)\tau^0 + g_1(n)\tau$ produces linearly independent $v_1, v_2 \in V(\hat{L})$. Then v_1^2, v_2^2 must be linearly independent as well, and hence form a basis of $V(\hat{L}^{\otimes 2})$. This basis is the image of u_1^2, u_2^2 under the map given by (6), and hence this map is onto (so the kernel must then have dimension $3-2 = 1$). $\square$

LEMMA 5. With notations as in Lemma 4, if L is irreducible then

1. $\hat{L}$ is irreducible

2. $\hat{L}^{\otimes 2}$ is irreducible

PROOF. The first item follows from Definitions 7, 8 and the assumption that L is irreducible. For the second item, assume that $\tau - s$ is a right-hand factor of $\hat{L}^{\otimes 2} = \tau^2 - r^2$. Then $r^2 = s\tau(s)$. Suppose that s is not a square, then take a point $p \in \mathbb{C}$ with maximal real part, for which s has a root or pole at p of odd order. Then $s\tau(s)$ has a root or pole at p with odd order, contradicting $r^2 = s\tau(s)$. Hence s is a square in $\mathbb{C}(n)$, say $s = t^2$. So $r = \pm t\tau(t)$, and after possibly multiplying s by $\sqrt{-1}$ we have $r = -t\tau(t)$. But then $\tau - t$ is a right-hand factor of $\hat{L} = \tau^2 + r$, contradicting item 1. $\square$

3. ALGORITHM FINDLIOUVILLIAN

For explanations and comments on the steps in the algorithm, see Section 4.

Algorithm FindLiouvillian

Input: $L \in \mathbb{C}[n][\tau]$ a second order, irreducible, homogeneous difference operator. Write $L = a_2(n)\tau^2 + a_1(n)\tau + a_0(n)$.

Output: A two-term difference operator, $\hat{L}$, with a gauge transformation from $\hat{L}$ to L , if it exists.

1. If $a_1 = 0$ then return $\hat{L} = L$ and stop.
2. Let $u(n)$ be an indeterminate function. Impose the relation $Lu(n) = 0$, i.e.

$$u(n+2) = -\frac{1}{a_2(n)}(a_0(n)u(n) + a_1(n)u(n+1)). \quad (7)$$

3. Let $d = \det(L) = a_0/a_2$. Let R be a non-zero rational solution of

$$L_T := L^{\otimes 2} \otimes (\tau + 1/d),$$

if such solution exists, else return NULL and stop. A formula for L_T is $d(n+2)d(n+1)d(n)c_3(n)\tau^3 - d(n+1)d(n)c_2(n)\tau^2 + d(n)c_1(n)\tau - c_0(n)$ where the c_i are from (4).

4. Let g be an indeterminate and let $v(n) = gu(n) + u(n+1)$. Compute $d_0, d_1, d_2 \in \mathbb{C}(n)[g]$ such that

$$v(n)^2 = d_0u(n)^2 + d_1u(n+1)^2 + d_2u(n+2)^2. \quad (8)$$

(To compute d_0, d_1, d_2 first substitute Equation (7) into Equation (8).)

5. Let S denote a non-zero solution of $\tau + d$, so $\tau(S) = -d(n)S$ and $\tau^2(S) = d(n+1)d(n)S$. Substitute the following

$$\begin{aligned} u(n)^2 &= R(n)S(n) \\ u(n+1)^2 &= -R(n+1)d(n)S(n) \\ u(n+2)^2 &= R(n+2)d(n+1)d(n)S(n) \end{aligned} \quad (9)$$

into Equation (8) to get $v(n)^2 = S(n)A$ for some $A \in \mathbb{C}(n)[g]$.

6. Solve $A = 0$ for g and choose one solution. A is a quadratic equation so this solution is in $\mathbb{C}(n)$ or in a quadratic extension of $\mathbb{C}(n)$. If $g \notin \mathbb{C}(n)$ then return an error message and stop.
7. Return $\hat{L}$ as well as the transformation $V(\hat{L}) \rightarrow V(L)$:

$$\hat{L} = \tau^2 + b(n)\frac{\tau(\delta)}{\delta} \quad (10)$$

$$u(n) = \frac{1}{\delta}((g(n+1) - a(n))v(n) - v(n+1)) \quad (11)$$

where $a(n) = a_1/a_2$, $b(n) = a_0/a_2$, $\delta = g(n)g(n+1) - g(n)a(n) + b(n)$, and $v(n)$ denotes a solution of $\hat{L}$.

REMARK 2. The formula for the gauge transformation given in Step 7 was found by computing the inverse of the gauge transformation $v(n) = gu(n) + u(n+1)$ introduced in Step 4 (where the value of g is computed in Steps 5 and 6).

EXAMPLE 2. Consider $nu(n+2) - u(n+1) - (n^2-1)(2n-1)u(n) = 0$. By downloading FindLiouvillian [16] and running it on $n\tau^2 - \tau -$

$(n^2 - 1)(2n - 1)$ one obtains:

$$\begin{aligned} d &= -(n^2 - 1)(2n - 1)/n \\ L_T &= n(n + 3)(2n + 3)(n + 1)^2 \tau^3 - \\ &\quad n(n + 2)(2n^3 + 3n^2 - n + 1)\tau^2 - \\ &\quad (n + 2)(n + 1)(2n^3 + 3n^2 - n + 1)\tau + \\ &\quad n(n + 2)(n - 1)(n + 1)(2n - 1) \end{aligned}$$

(denominators were multiplied away by taking the primitive part)

$$R = \frac{1}{n}, \quad A = \frac{1}{n} \cdot (g^2 + (3n - 2)g + (2n - 1)(n - 1))$$

$$g = 1 - n, \quad \delta = 1 - n^2$$

and finally:

$$\hat{L} = \tau^2 - (2n - 1)(n + 2), \quad u(n) = \frac{1}{n}v(n) + \frac{1}{n^2 - 1}v(n + 1).$$

The algorithm is explained in Section 4 below. As mentioned in the Introduction, the main point is to show that the hypergeometric solution of $L^{\otimes 2}$ corresponds to a rational solution of $L^{\otimes 2} \otimes (\tau + 1/\det(L))$ as this is what allows us to eliminate the combinatorial aspect of the algorithm.

4. EXPLANATION

Idea: Assume that $L = \tau^2 + a_1\tau + a_0$, $a_1 \neq 0$ is Liouvillian and irreducible, then there exists a gauge transformation $L \rightarrow \hat{L}$ for some $\hat{L}$ of the form $\tau^2 + r$ (see Theorem 2). $\dim(V(L^{\otimes 2})) = 3$ and $\dim(V(\hat{L}^{\otimes 2})) = 2$ so the transformation $L^{\otimes 2} \rightarrow \hat{L}^{\otimes 2}$ has a 1-dimensional kernel corresponding to a right hand factor of $L^{\otimes 2}$ (i.e. $L^{\otimes 2}$ has a hypergeometric solution, namely RS in Step 5). This gives us the gauge transformation from $V(L)$ to $V(\hat{L})$. In order that Step 3 only needs to search for a solution $\in \mathbb{C}(n)$ (which is easier than computing a more general hypergeometric solution) we work with the symmetric product of $L^{\otimes 2}$ and $\tau - 1/\det(L)$ and use Theorem 3.

4.1 Algorithm Step 3

Let L be from **Input** and let $\hat{L}$ be a two-term operator.

REMARK 2. A symmetric product of a second order two-term operator with a first order difference operator is again a second order two-term operator. It follows that we may disregard the symmetric product, i.e. we need only search for a gauge transformation.

LEMMA 6. Let $A_1, A_2, G_2 \in \mathbb{C}(n)[\tau]$, $\text{ord}(A_1) = 3, \text{ord}(A_2) = 2$, and assume that $G_2(V(A_1)) = V(A_2)$, i.e. $V(A_1) \xrightarrow{G_2} V(A_2)$. Then A_1 has a first order right hand factor, as well as a second order left hand factor that is gauge equivalent to A_2 .

PROOF. $V(\text{GCRD}(A_1, G_2)) = V(A_1) \cap V(G_2) = \ker(G_2: V(A_1) \rightarrow V(A_2))$ which has dimension $3 - 2 = 1$ and so A_1 has a first order right hand factor, $L_1 = \text{GCRD}(A_1, G_2)$. Write $A_1 = L_2 L_1$, $G_2 = \tilde{G} L_1$ then $\tilde{G}: V(L_2) \rightarrow V(A_2)$ shows that L_2 is gauge equivalent to A_2 . $\square$

For the theorem below we now substitute $L^{\otimes 2}, \hat{L}^{\otimes 2}$ for A_1, A_2 , respectively, from the preceding Lemma.

THEOREM 3. Let $\hat{L} = r_2\tau^2 + r_0$ and let $L = a_2\tau^2 + a_1\tau + a_0$, $a_i \neq 0$. Suppose there is a gauge transformation $G: V(L) \rightarrow V(\hat{L})$ then by Lemma 4 there is $G_2: V(L^{\otimes 2}) \rightarrow V(\hat{L}^{\otimes 2})$. Let $L_1 = \text{GCRD}(G_2, L^{\otimes 2})$ (which has order 1 by Lemma 6) and write $L^{\otimes 2} = L_2 L_1$. Then

$$\det(L_1) \stackrel{\text{SE}}{=} -\det(L).$$

PROOF. 1. $\det(L) \stackrel{\text{SE}}{=} \det(\hat{L})$, see Lemma 2.

2. $\det(L^{\otimes 2}) \stackrel{\text{SE}}{=} \det(L)^3$, see the formula for $L^{\otimes 2}$ in Lemma 3.

3. $\det(\hat{L}^{\otimes 2}) = -\det(\hat{L})^2 \stackrel{\text{SE}}{=} -\det(L)^2$, see Lemma 3 with $a_1 = 0$ and Item 1, respectively.

4. $\det(L^{\otimes 2}) = \det(L_2 L_1) \stackrel{\text{SE}}{=} \det(L_2) \det(L_1)$.

5. L_2 is gauge equivalent to $\hat{L}^{\otimes 2}$, see Lemma 6.

6. $\det(L_2) \stackrel{\text{SE}}{=} \det(\hat{L}^{\otimes 2}) = -\det(\hat{L})^2 \stackrel{\text{SE}}{=} -\det(L)^2$, see Items 5,3,1.

7. $\det(L_1) \stackrel{\text{SE}}{=} -\det(L)$, see Items 4,2,6.

$\square$

COROLLARY 1. Let $L, \hat{L}, G$ be as in Theorem 3 so that $L^{\otimes 2} = L_2 L_1$ then there exists a rational solution of $L^{\otimes 2} \otimes (\tau + 1/\det(L))$.

Step 3 computes this rational solution. The solution is rational because the first order factor has determinant shift equivalent to 1 (the solution is $r(n)$ from Remark 1). Lemma 5 combined with Lemma 6 shows that if L is irreducible, then L_2 in the corollary is irreducible, which implies that the rational solution is unique (up to constant multiples).

4.2 Algorithm Step 4

LEMMA 7. If $T_1: g_1\tau + g_0$ with $g_1 \neq 0$ defines a gauge transformation from L to a two-term operator then $T_2: \tau + g_0/g_1$ is also a gauge transformation from L to a two-term operator.

The two transformations differ by the term product $u \mapsto g_1 u$ and so the Lemma's claim follows from Remark 2. (The case $g_1 = 0$ defines a term product.)

4.3 Algorithm Step 5

Equation (8) defines the map $L^{\otimes 2} \rightarrow \hat{L}^{\otimes 2}$ and RS is in the kernel. Both $u(n)^2$ and RS satisfy $L^{\otimes 2}$, see steps 2 and 3. (Recall that RS is the hypergeometric solution of $L^{\otimes 2}$ that we computed using Corollary 1.)

4.4 Algorithm Step 6

It can be proven that if there exists a gauge transformation from $L \in \mathbb{C}(n)[\tau]$ to an operator of the form $\tau^2 + \alpha$ where α is algebraic over $\mathbb{C}(n)$ then there also exists a gauge transformation $G: L \rightarrow \tilde{L} = \tau^2 + \tilde{\alpha}$ with $G, \tilde{L} \in \mathbb{C}(n)[\tau]$. Note that if $L \in \mathbb{C}(n)[\tau]$, $C \subset \mathbb{C}$ then an algebraic extension of C may occur, see the following example.

EXAMPLE 3. $L_1 = \tau^2 - \tau - n^2 + 1$ and $L_2 = \tau^2 - (n + i)(n + 1 - i)$. are gauge equivalent with $L_1 \in \mathbb{Q}(n)[\tau]$ and $L_2 \in \mathbb{Q}(n)[\tau, \sqrt{-1}]$. Both $\frac{1}{n-i}\tau + 1$, which sends L_2 to L_1 , and its inverse are $\in \mathbb{C}(n)[\tau]$.

5. ADDITIONAL WORK

It is interesting to note that if we introduce a minus sign in Step 3 of Algorithm FindLiouvillian, setting $d := -a_0/a_2$, then the algorithm turns into the so-called eigenring method (Section 4 in [4]). Specifically, the $\tau + g$ computed in the algorithm (this time, use both solutions of A) will then be right-hand factors of L (this is not a complete algorithm for factoring L , it works if L has precisely two first order right-hand factors in $\mathbb{C}(n)[\tau]$).

To generalize our algorithm to n 'th order, we need to compute a gauge transformation from L to $L \otimes (\tau - \zeta_n)$ where ζ_n is an n 'th root of unity. So we need to find or write an implementation that can compute gauge transformations while allowing algebraic numbers in the input.

6. REFERENCES

- [1] N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences. World-Wide Web URL www.research.att.com/~njas/sequences/
- [2] S. A. Abramov. Rational solutions of linear difference and q -difference equations with polynomial coefficients. In *(Russian) Programirovanie*, **6**: 3-11, 1995; translation in *Program. Comput. Software* **21**, **6**: 273-278, (1995).
- [3] S.A. Abramov, M.A. Barkatou, D.E. Khmelnov. On m -Interlacing Solutions of Linear Difference Equations, LNCS, **5743**, p. 1-17 (2009).
- [4] M. A. Barkatou. Rational Solutions of Matrix Difference Equations: The Problem of Equivalence and Factorization, ISSAC' 1999, p. 277–282, (1999).
- [5] M. van Hoeij. Finite singularities and hypergeometric solutions of linear recurrence equations. In *Journal of Pure and Applied Algebra*, **139**: 109-131, 1999.
- [6] T. Cluzeau, M. van Hoeij. Computing Hypergeometric Solutions of Linear Recurrence Equations. In *Applicable Algebra in Engineering, Communication and Computing*, **17**(2): 83-115, 2006.
- [7] M. Petkovšek. Hypergeometric solutions of linear recurrences with polynomial coefficients. In *Journal of Symbolic Computation*, **14**(2-3): 243-264, 1992.
- [8] R. Feng, M. F. Singer, M. Wu. Liouvillian solutions of difference-differential equations. To appear in *Journal of Symbolic Computation*, 2009.
- [9] M. Petkovšek, H. S. Wilf, D. Zeilberger. $A = B$. With a foreword by Donald E. Knuth. A. K. Peters, Ltd., Wellesley, MA, 1996.
- [10] M. van der Put, M. F. Singer. Galois theory of difference equations. In *Lecture Notes in Mathematics*, vol. **1666**, Springer, Berlin, 1997.
- [11] P.A. Hendriks An algorithm for determining the difference Galois group for second order linear difference equations. In *Journal of Symbolic Computation*, **26**: 445–462 1998.
- [12] P. A. Hendriks, M. F. Singer. Solving difference equations in finite terms. In *Journal of Symbolic Computation*, **27**: 239-259, 1999.
- [13] M. F. Singer. Testing reducibility of linear differential operators: a group theoretic perspective. In *Applicable Algebra in Engineering, Communication and Computing*, **7**(2): 77-104, 1996.
- [14] Y. Cha, M. van Hoeij. Liouvillian solutions of irreducible linear difference equations. In *ISSAC '09: Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, Pages: 87-94, 2009.
- [15] G. Levy. Solutions of second order recurrence relations. Ph.D. dissertation, Florida State University, 2010. Text and implementations available at: <http://www.math.fsu.edu/~glevy/implementation>
- [16] M. van Hoeij, G. Levy. FindLiouvillian implementation. <http://www.math.fsu.edu/~hoeij/files/FindLiouvillian>

Solving Recurrence Relations using Local Invariants

Yongjae Cha
Department of Mathematics
Florida State University
Tallahassee, FL 32306, USA
ycha@math.fsu.edu

Mark van Hoeij*
Department of Mathematics
Florida State University
Tallahassee, FL 32306, USA
hoeij@math.fsu.edu

Giles Levy
Department of Mathematics
Florida State University
Tallahassee, FL 32306, USA
glevy@math.fsu.edu

ABSTRACT

The goal in this paper is to find closed form solutions for linear recurrence equations, by transforming an input equation L to an equation L_s with known solutions. The main problem is how to find a solved equation L_s to which L can be reduced. We solve this problem by computing local data at singularities, data that remains invariant under the transformations used.

Categories and Subject Descriptors

G.2.1 [Combinatorics]: [Recurrences and difference equations]; I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—Algebraic algorithms

General Terms

Algorithms

1. INTRODUCTION

Let $L = a_n\tau^n + \dots + a_0\tau^0$ denote a linear difference operator. Here τ denotes the shift operator $\tau(f(x)) = f(x+1)$ and the a_i are rational functions in x (after multiplying away the denominators, we may assume that the a_i are in $\mathbb{C}[x]$). Now L corresponds to the recurrence relation $a_n u(x+n) + \dots + a_0 u(x+0) = 0$.

This paper presents a new approach to finding solutions to linear recurrence relations with polynomial coefficients. The general approach is to transform an equation to a previously solved equation.

Given two linear recurrence operators L_1 and L_2 of the same order with coefficients from $\mathbb{C}[x]$, we give an algorithm which finds (if it exists) a map, in terms of so-called gauge transformations and term transformations, which sends the solutions of L_1 bijectively to L_2 . Hence, if closed form solutions of L_1 are known, one obtains closed form solutions of L_2 . Due to the fact that in the literature many special functions are defined by recurrence relations involving

parameters (like orthogonal polynomials, Bessel functions of the first and second kind), we consider the following refined version: given a linear recurrence, extract information in terms of the free parameters such that any term and/or gauge transformation keeps this expression unchanged; this information is also called invariant data. In this way, one can construct a table of special functions together with its defining recurrence relations and its invariant data (w.r.t. the free parameters). One can efficiently determine for a given recurrence relation (whose solutions are unknown) the possible recurrence candidates together with the appropriate choices of the free parameters. Then for the derived choices one can check step by step if a transformation exists. This mechanism has been worked out in detail for linear recurrences of order 2.

In our previous paper [7] the table consisted of a single equation, namely $\tau^n + b$, but this equation contained a parameter $b \in \mathbb{C}(x)$, so this one equation represents an infinite set of equations, parametrized by $b \in \mathbb{C}(x)$. Now before we can compute, if it exists, a transformation between the input equation L and an equation of the form $\tau^n + b$, we first have to compute the unknown $b \in \mathbb{C}(x)$. Most of [7] is devoted to finding a finite list of candidates for b .

In [7], the invariant data was the finite singularities. In this paper, in addition to finite singularities, we shall also use local data at infinity (generalized exponents). The reason for using both in this paper is the following: The more solved equations we add to the table, the stronger the solver will become; however, we can only add equations to the table if the parameters in those equations can be computed from the invariant data that we compute.

This means that the more invariant data we compute, the stronger we can make the solver. So besides using the implementation for finite singularities that was used in [7] (and in earlier papers [6]) we also implemented an algorithm to compute generalized exponents at the point $x = \infty$.

The mathematics of these generalized exponents has been treated in [11]¹, and computationally, one can view generalized exponents as a portion of formal solutions, a topic for which algorithms have been developed [4]. Therefore, one can compute generalized exponents by implementing, as we did, a portion of an algorithm [4] to compute formal solutions. Although this mathematics is known [4, 11], it is not widely known, and so, for the convenience of the reader, we included an appendix on this topic. What is new in this paper is not the concept of generalized exponents itself, but

*Supported by NSF grant 0728853

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

¹the characteristic classes in [11] are the minimal polynomials of what are called “generalized exponents” in this paper

rather the way this concept is used. We use it not to compute local solutions (formal solutions containing truncated power series), but global solutions.

2. PRELIMINARIES AND DEFINITIONS

For further treatment on topics in this section see [6], [7], [8], [9], [10], and [14].

DEFINITION 2.1. [*Galois Theory of Difference Equations*, Example 1.3 [14]] Define $S = \mathbb{C}^{\mathbb{N}}/\sim$ where $s_1 \sim s_2$ if there exists $N \in \mathbb{N}$ such that, for all $x > N$, $s_1(x) = s_2(x)$.

LEMMA 2.2. A unit is a sequence in S that is invertible, i.e. a sequence that only has finitely many zeros.

DEFINITION 2.3. $V(L)$ refers to the solution space of the operator L , i.e. $V(L) = \{u \in S \mid Lu = 0\}$, where S is as in Definition 2.1.

Remark $V(L)$ forms a subspace of S over $\mathbb{C}$.

THEOREM 2.4. [*'A=B' Theorem 8.2.1 [13]*] Let $L = \sum_{k=0}^n a_k \tau^k$ be a linear difference operator of order n on S . If a_0 and a_n are units, then $\dim(V(L)) = n$.

We view $\mathbb{C}(x)$ as a subset of S (see Section 8.2 in [13]) so the theorem applies to $L \in \mathbb{C}(x)[\tau]$ with $a_0, a_n \neq 0$.

DEFINITION 2.5. Two operators L_1, L_2 in $\mathbb{C}(x)[\tau]$ are called gauge equivalent if and only if there exists an operator $G \in \mathbb{C}(x)[\tau]$ such that $G(V(L_1)) = V(L_2)$ and L_1, L_2 have the same order. This G is a gauge transformation $L_1 \rightarrow L_2$ and we denote gauge equivalence by $L_1 \sim_g L_2$.

If $L_1 \sim_g L_2$, and $G : V(L_1) \rightarrow V(L_2)$ is a gauge transformation, then the inverse gauge transformation $V(L_2) \rightarrow V(L_1)$ can be computed with the extended Euclidean Algorithm as follows: Let GCRD denote greatest common right divisor. Since $G : V(L_1) \rightarrow V(L_2)$ is a bijection, the kernel of G on $V(L_1)$ is $\{0\}$ and thus $V(G) \cap V(L_1) = \{0\}$. So, $V(\text{GCRD}(G, L_1)) = \{0\}$ and $\text{GCRD}(G, L_1) = 1$. There exists $S, T \in \mathbb{C}(x)[\tau]$ such that $SG + TL_1 = 1$ by the extended Euclidean Algorithm for $\mathbb{C}(x)[\tau]$. Then $SG \equiv 1 \pmod{L_1}$. So $SG : V(L_1) \rightarrow V(L_1)$ is the identity map and $S : V(L_2) \rightarrow V(L_1)$ is the inverse of G .

DEFINITION 2.6. Let $L_1, L_2 \in \mathbb{C}(x)[\tau]$. The symmetric product of L_1 and L_2 written $L_1 \otimes L_2$ is defined as the monic operator $L \in \mathbb{C}(x)[\tau]$ of minimal order such that $L(u_1 u_2) = 0$ for all u_1, u_2 with $u_1 \in V(L_1)$ and $u_2 \in V(L_2)$. For the case $L_2 = \tau - r$ with $r \in \mathbb{C}(x)$ we call $\otimes L_2$ a term transformation which is an action on $\mathbb{C}(x)[\tau]$.

The formula for a term transformation is

$$L \otimes (\tau - r) = \frac{1}{b_n} \sum_{i=0}^n b_i \tau^i \quad (1)$$

where $b_n = a_n$ and $b_i(x) = a_i(x) \prod_{j=i}^{n-1} \tau^j(r(x))$.

Given a series of gauge and term transformations from one operator to another, the following theorems reduce the problem of finding those transformations to that of finding exactly one gauge and one term transformation.

THEOREM 2.7. [Theorem 3.3 [8]] Let $s_1, \dots, s_m$ be some combination of gauge transformations and term transformations. A transformation $L_1 \xrightarrow{s_1 \circ \dots \circ s_m} L_2$ can be written $L_1 \xrightarrow{t_2 \circ t_1} L_2$ for some gauge transformation t_1 and some term transformation t_2 .

DEFINITION 2.8. $L_1 \xrightarrow{t_2 \circ t_1} L_2$, for some gauge transformation t_1 and some term transformation t_2 , will be called a GT-transformation.

DEFINITION 2.9. Let $r(x) = cp_1(x)^{e_1} \dots p_j(x)^{e_j} \in C(x)$ with $C \subseteq \mathbb{C}$. Let the $e_i \in \mathbb{Z}$, let the $p_i(x)$ be irreducible in $C[x]$, and let $s_i \in C$ equal the sum of the roots of $p_i(x)$. $r(x)$ is said to be in shift normal form if $-\deg(p_i(x)) < \text{Re}(s_i) \leq 0$, for $i = 1, \dots, j$. We denote $\text{SNF}(r(x))$ as the shift normalized form of $r(x)$ which is obtained by replacing each $p_i(x)$ by $p_i(x + k_i)$ for some $k_i \in \mathbb{Z}$ such that $p_i(x + k_i)$ is in shift normal form.

Remark: If $r \in \mathbb{Q}(x)$ then factoring in $\mathbb{Q}[x]$ is easier than in $\mathbb{C}[x]$. Take $C = \mathbb{Q}$ in this case. Also, $\text{SNF}(r(x))$ is unique up to choice of $C \subseteq \mathbb{C}$.

DEFINITION 2.10. Let $L = a_2 \tau^2 + a_1 \tau + a_0$, $a_2 \neq 0$. The determinant of L , $\det(L)$, is defined to be a_0/a_2 . (It is the determinant of the 'companion matrix' in the corresponding matrix representation of L .)

THEOREM 2.11. [Theorem 3.4 [8]] Let L_1, L_2 each have order 2 and the leading and trailing coefficient of L_1 be non-zero. If $L_1 \xrightarrow{t_2 \circ t_1} L_2$ for some gauge transformation t_1 and some term transformation t_2 then there exists a gauge transformation $L_1 \otimes (\tau - r) \rightarrow L_2$, where

$$r = \pm \sqrt{\text{SNF}(\det(L_2)/\det(L_1))}.$$

DEFINITION 2.12. It is in the context of Definition 2.8 (and Theorem 2.11) that we say that L_2 can be reduced to L_1 .

We provide an algorithm from [8] that finds such a reduction if it exists.

Algorithm Find GT-Transformation:

Input: $L_1, L_2 \in \mathbb{C}[x][\tau]$ linear difference operators of order 2.

Output: Operator of the form $H(x)(c_1(x)\tau + c_0(x))$ mapping $V(L_1)$ to $V(L_2)$ if it exists.

1. Calculate $\hat{r} = \text{SNF}(\det(L_2)/\det(L_1))$.
2. If $\hat{r}$ is a square in $\mathbb{C}(x)$ then let $r = \sqrt{\hat{r}}$ else return 'FAIL' and stop.
3. Calculate $L_{\text{neg}} = L_1 \otimes (\tau - r)$ and $L_{\text{pos}} = L_1 \otimes (\tau + r)$.
4. Compute a gauge transformation, $c_1(x)\tau + c_0(x)$, between L_{neg} and L_2 (see [3] or [8]).
 - (a) If a gauge transformation exists then return $H(x) \cdot (c_1(x)\tau + c_0(x))$ and exit, where $H(x)$ is a solution of $\tau - r$.
 - (b) If no gauge transformation exists then go to Step 5.
5. Compute a gauge transformation, $c_1(x)\tau + c_0(x)$, between L_{pos} and L_2 .

- (a) If a gauge transformation exists then return $H(x) \cdot (c_1(x)\tau + c_0(x))$ and exit, where $H(x)$ is a solution of $\tau + r$.
- (b) If no gauge transformation exists return 'FAIL.'

EXAMPLE 2.13. Here we will check the above algorithm with operators in Example 6.2. Let $L_1 = -\frac{1}{3}(2+x)\tau^2 + (2 + \frac{4}{3}x)\tau - 1 - x$ and $L_2 = (x+4)\tau^2 + (-20-8x)\tau + (12x+12)$. Then $\hat{r} = 4$ and $r = 2$. By computing gauge transformation from $L_1 \otimes (\tau - 2)$ to L_2 we get $\frac{1}{x+2}(-\tau + 3)$. Thus the algorithm returns $2^x(\frac{1}{x+2}(-2\tau + 3))$. (A 2 appeared in front of τ because $\tau 2^x = 2^x 2\tau$.)

3. FINITE SINGULARITIES

Local data of a difference operator are valuation growths at finite singularities in $\mathbb{C}/\mathbb{Z}$ and generalized exponents at the point of infinity. In this section, we will review the definition and an invariance property (Theorem 3.4) of valuation growths from [7], [9].

Let $L = a_n\tau^n + \dots + a_0\tau^0 \in \mathbb{C}(x)[\tau]$. After multiplying L on the left by a suitable element of $\mathbb{C}(x)$, we may assume that the a_i are in $\mathbb{C}[x]$ and $\gcd(a_0, \dots, a_n) = 1$.

DEFINITION 3.1. Let $L = a_n\tau^n + \dots + a_0\tau^0$ with $a_i \in \mathbb{C}[x]$. $q \in \mathbb{C}$ is called a problem point of L if q is a root of the polynomial $a_0(x)a_n(x-n)$. $p \in \mathbb{C}/\mathbb{Z}$ is called a finite singularity of L if L has a problem point in p (i.e. $p = q + \mathbb{Z}$ for some problem point q).

DEFINITION 3.2. Let $u(x) \in \mathbb{C}(x)$ be a non-zero meromorphic function. The valuation growth $g_p(u)$ of $u(x)$ at $p = q + \mathbb{Z}$ is

$$\liminf_{n \rightarrow \infty} (\text{order of } u(x) \text{ at } x = n + q) \\ - \liminf_{n \rightarrow \infty} (\text{order of } u(x) \text{ at } x = -n + q).$$

Define the set of valuation growths of L at p as

$$\bar{g}_p(L) = \{g_p(u) \mid u \neq 0 \text{ meromorphic solution of } L\} \subset \mathbb{Z}.$$

Note: The definition of valuation growths in [7] was longer, but using ideas from [2], the two definitions can be shown to be equivalent.

DEFINITION 3.3. Let L be a difference operator and $p \in \mathbb{C}/\mathbb{Z}$ be a finite singularity of L . If $\bar{g}_p(L)$ has more than one element then p is called an essential singularity.

THEOREM 3.4. [Theorem 1 in [7]] If L_1 and L_2 are gauge equivalent then $\bar{g}_p(L_1) = \bar{g}_p(L_2)$ for every $p \in \mathbb{C}/\mathbb{Z}$.

Let $\tilde{L} = L \otimes (\tau - a)$ for some $a \in \mathbb{C}(x)$ and let $\bar{g}_p(\tau - a) = \{d\}$ for some $d \in \mathbb{Z}$ and $p \in \mathbb{C}/\mathbb{Z}$. Then $\bar{g}_p(\tilde{L}) = \{n + d \mid n \in \bar{g}_p(L)\}$. Therefore term transformations do not preserve $\bar{g}_p(L)$ but they do preserve $d_p(L) := \max \bar{g}_p(L) - \min \bar{g}_p(L)$. We define the set

$$\text{Val}(L) = \{[p, d_p(L)] \mid p \in \mathbb{C}/\mathbb{Z} \text{ essential singularity of } L\}.$$

We compute $\text{Val}(L)$ (see our table in Section 5) because it is data that is invariant under GT -transformations.

4. GENERALIZED EXPONENTS

Generalized exponents are local data at the point of infinity. A mathematically equivalent concept has been discussed in [4], [6], and [11]. The main techniques for computing generalized exponents are indicial equations, Newton τ -polygons, and Newton Δ -polygons of a difference operator, the same as in the computation of formal solutions in [4], [5]. Computing generalized exponents in an unramified case is also explained in [6, Section 5]. Here we will define generalized exponents of a difference operator and introduce Theorem 4.4, which is one of the key tools that the algorithm given in Section 7 uses.

Denote $t = 1/x$ and let

$$K = \mathbb{C}((t)).$$

Define the following ring of difference operators:

$$\mathbf{D} := K[\tau].$$

Now $\mathbb{C}(x) \subset K$ and the action of τ on $\mathbb{C}(x)$ can be extended to an action on K .

$$\tau(t) = \tau\left(\frac{1}{x}\right) = \frac{1}{x+1} = \frac{t}{1+t} = t - t^2 + \dots \in K.$$

The field K has a natural valuation $v : K \rightarrow \mathbb{Z} \cup \{\infty\}$ where $v(0) := \infty$ and

$$v(c_n t^n + c_{n+1} t^{n+1} + \dots) = n \text{ if } c_n \neq 0.$$

Let $\Delta = \tau - 1$, then $\mathbf{D} = K[\tau] = K[\Delta]$. Let $L \in \mathbf{D}$ and write $L = \sum_{i=0}^d a_i \Delta^i$. Now we extend the definition of v to $\mathbf{D}$ as follows

$$v(L) := \min\{v(a_i) + i \mid i = 0, \dots, d\}.$$

Note that this $v : \mathbf{D} \rightarrow \mathbb{Z} \cup \{\infty\}$ still satisfies the properties of a valuation:

- (i) $v(L) = \infty \iff L = 0$,
- (ii) $v(L_1 + L_2) \geq \min\{v(L_1), v(L_2)\}$, (equality when $v(L_1) \neq v(L_2)$)
- (iii) $v(L_1 L_2) = v(L_1) + v(L_2)$ (follows from Corollary 9.1 in the Appendix).

LEMMA 4.1. Let $L \in K[\tau]$. There exists a polynomial P such that for every $n \in \mathbb{Z}$ we have

$$L(t^n) = P(n)t^{n+v(L)} + \dots \quad (2)$$

where the dots refer to terms of valuation $> n + v(L)$.

PROOF. Let $\text{tc}(f)$ be the trailing coefficient of $f \in \mathbb{C}((t))$. $\Delta^i(t^n) = P_i(n)t^{n+i} + \dots$ where $P_i(n) = (-1)^i n(n+1) \dots (n+i-1)$ and $a_i \Delta^i(t^n) = P_i(n)\text{tc}(a_i)t^{n+i+v(a_i)} + \dots$. Let $M = \{i \in \mathbb{Z} \mid v(a_i) + i = v(L)\}$ then

$$L(t^n) = \sum_{i \in M} P_i(n)\text{tc}(a_i)t^{n+v(L)} + \dots$$

Then $P(n) = \sum_{i \in M} P_i(n)\text{tc}(a_i)$. $\square$

DEFINITION 4.2. Ind_L , the indicial equation² of L , is the polynomial, $P(n)$, constructed in the proof of lemma 4.1.

²for further discussion of the indicial equation, see the Appendix

For each $r \in \mathbb{N}$ we denote $K_r = \mathbb{C}((t^{1/r}))$. The algebraic closure of K is $\bar{K} = \bigcup_{r \in \mathbb{N}} K_r$. Define the action of τ on K_r :

$$\begin{aligned} \tau(t^{\frac{1}{r}}) &= t^{\frac{1}{r}}(1+t)^{-\frac{1}{r}} \\ &= t^{\frac{1}{r}}(1 - \frac{1}{1!} \frac{1}{r} t + \frac{1}{2!} \frac{1}{r} (\frac{1}{r} + 1) t^2 \\ &\quad - \frac{1}{3!} \frac{1}{r} (\frac{1}{r} + 1) (\frac{1}{r} + 2) t^3 + \dots) \in K_r. \end{aligned} \quad (3)$$

Since we have defined the action of τ on K_r , we can now apply the formula for the term transformation in Equation (1) to $K_r[\tau]$. Let $\tilde{G}_r$ and E_r be the following subgroup and subset, respectively, of K_r^* .

$$\tilde{G}_r = \left\{ a \in K_r^* \mid a = 1 + \sum_{i=r+1}^{\infty} a_i t^{i/r}, a_i \in \mathbb{C} \right\}$$

$$E_r = \left\{ a \mid a = ct^v(1 + \sum_{i=1}^r a_i t^{i/r}), a_i \in \mathbb{C}, c \in \mathbb{C}^*, v \in \frac{1}{r}\mathbb{Z} \right\}$$

Now E_r is a set of representatives for $K_r^*/\tilde{G}_r$. The composition of the natural maps $K_r^* \rightarrow K^*/\tilde{G}_r \rightarrow E_r$ defines a natural map

$$\text{Trunc} : K_r^* \rightarrow E_r.$$

Let

$$G_r = \{a \in K_r^* \mid a = 1 + \frac{m}{r}t + \sum_{i=r+1}^{\infty} a_i t^{i/r}, a_i \in \mathbb{C}, m \in \mathbb{Z}\}.$$

If $a, b \in E_r$ then we say $a \sim_r b$ when $a/b \in G_r$.

Note: $a \sim_r b$ if and only if $a_r \equiv b_r \pmod{\frac{1}{r}\mathbb{Z}}$ with a_r as in the definition of E_r , and all the other parts of a (the numbers $c, v, a_1, \dots, a_{r-1}$) are the same for b .

DEFINITION 4.3. Let $a \in E_r$ for some $r \in \mathbb{N}$. We say that a is a generalized exponent of L with multiplicity m if and only if 0 is a root of $\text{Ind}_{\tilde{L}}$ with multiplicity m where $\tilde{L} = L \otimes (\tau - \frac{1}{a})$. We denote $\text{genexp}(L)$ as the set of generalized exponents of L .

THEOREM 4.4. Suppose $L_1 \sim_g L_2$, then for each $a \in \text{genexp}(L_1)$ there exists $b \in \text{genexp}(L_2)$ such that $a \sim_r b$. (Where r is minimal with $a \in E_r$.)

PROOF. Let $G \in \mathbb{C}(x)[\tau]$ be a gauge transformation from L_1 to L_2 and $a \in E_r$ for some $r \in \mathbb{N}$ be a generalized exponent of L_1 . One can verify that $G_A := A \cdot G \cdot A^{-1}$ is an element of $\mathbb{C}(x, a)[\tau]$ and G_A is a bijection from $V(L_1 \otimes (\tau - 1/a))$ to $V(L_2 \otimes (\tau - 1/a))$ where A is a solution of $\tau - 1/a$ (a basis of solutions of any operator in $\bar{K}[\tau]$ can be found in the universal extension of $\bar{K}$, see Section 6.2 of [14]). Since a is a generalized exponent of L_1 , $L_1 \otimes (\tau - 1/a)$ has 0 as a root of its indicial equation. Then it has a solution in K_r by Lemma 9.3 in the Appendix. By applying G_A , we find that $L_2 \otimes (\tau - 1/a)$ also has a solution in K_r . Then its indicial equation has a root in $\frac{1}{r}\mathbb{Z}$ by Lemma 9.3 in the Appendix. Let m/r be such a root for some $m \in \mathbb{Z}$. Then

$$L_2 \otimes (\tau - 1/a) \otimes (\tau - (1 + \frac{m}{r}t))$$

has 0 as a root of the indicial equation. Then by Equation 1, $(\tau - 1/a) \otimes (\tau - (1 + \frac{m}{r}t)) = \tau - \frac{1}{a}(1 + \frac{m}{r}t)$. Thus, $\text{Trunc}(a/(1 + \frac{m}{r}t))$ is a generalized exponent of L_2 . $\square$

The above theorem says generalized exponents mod $\sim_r$ are invariant under gauge transformations. Suppose L has order 2 , and let $\text{genexp}(L) = \{a_1, a_2\}$ and $\tilde{L} = L \otimes (\tau - \alpha)$ for some $\alpha \in K_r, r \in \mathbb{N}$. Then

$$\text{genexp}(\tilde{L}) = \{\text{Trunc}(\alpha a_1), \text{Trunc}(\alpha a_2)\}.$$

To obtain an expression that is invariant under the term transformations as well, we define the quotient of the generalized exponents.

DEFINITION 4.5. Suppose L has order 2 , and $\text{genexp}(L) = \{a_1, a_2\}$ such that $v(a_1) \geq v(a_2)$. If $v(a_1) > v(a_2)$ then we define the set of quotient of the two generalized exponents as $\text{Gquo} = \{\text{Trunc}(a_1/a_2)\}$. If $v(a_1) = v(a_2)$ then we define $\text{Gquo}(L) = \{\text{Trunc}(a_1/a_2), \text{Trunc}(a_2/a_1)\}$.

An example of computing a generalized exponent in unramified case is explained in [6]. An example follows:

EXAMPLE 4.6. $L_{WM} = (2n + 2\nu + 3 + 2x)\tau^2 + (2z - 4\nu - 4x - 4)\tau - 2n + 1 + 2\nu + 2x$ is an operator from the table in Section 5. Suppose $\text{genexp}(L_{WM}) = \{c_1 t_1^v(1 + a_1 t^{\frac{1}{2}} + a_2 t), c_2 t_2^v(1 + b_1 t^{\frac{1}{2}} + b_2 t)\}$. The slope of the Newton τ -polygon of L_{WM} is 0 and the corresponding Newton τ -polynomial is $2(t - 1)^2$. So $c_1 = c_2 = 1$ and $v_1 = v_2 = 0$ (the c_i, v_i are the values for the numbers c, v from the definition of E_r). Since $\text{Ind}_{L_{WM}} = 2z$, $\text{Ind}_{L_{WM}}$ has no root, that is, a_1 and b_1 are not 0 . So we need to calculate the Newton δ -polygon of L_{WM} . Then the slope of the Newton δ -polygon is $\frac{1}{2}$ and its Newton δ -polynomial is $2z + 2t^2$, which gives $a_1 = \sqrt{-z}$ and $b_1 = -\sqrt{-z}$. The indicial equation of both $L_{WM} \otimes (\tau - 1/(\sqrt{-z}t^{\frac{1}{2}}))$ and $L_{WM} \otimes (\tau - (1/\sqrt{-z}t^{\frac{1}{2}}))$ is $-64z(1 + 2z + 4\nu) + 256nz$, so the root of the indicial equation is $n = \frac{1}{4} + \frac{1}{2}z + \nu$. Thus, $a_2 = b_2 = -n$, $\text{genexp}(L_{WM}) = \{1 + \sqrt{-z}t^{\frac{1}{2}} - (\frac{1}{4} + \frac{1}{2}z + \nu)t, 1 - \sqrt{-z}t^{\frac{1}{2}} - (\frac{1}{4} + \frac{1}{2}z + \nu)t\}$, and $\text{Gquo} = \{1 - 2\sqrt{-z}t^{\frac{1}{2}} - 2zt, 1 + 2\sqrt{-z}t^{\frac{1}{2}} - 2zt\}$.

5. BASE EQUATIONS OF DEGREE OF 2

Many special functions satisfy recurrences w.r.t their parameters as in [1]. We use these recurrences as base equations in the table below. The table also contains the corresponding local data that we computed with our implementation. At the moment the table below contains a somewhat arbitrary set of base equations. It is easy to add more, and our goal is to do that in a systematic way. The table lists base equations with their known solution(s) and local data. Here B_I and B_J denote Bessel functions of the first kind, B_K and B_Y denote Bessel functions of the second kind, W_W denotes the Whittaker W function and W_M denotes the Whittaker M function.

$$1. L_{IK} = z\tau^2 + (2 + 2\nu + 2x)\tau - z$$

- Constants: z, ν
- Solutions = $\{B_I(\nu + x, z), B_K(\nu + x, -z)\}$
- $\text{Gquo} = \{-\frac{z}{4}t^2(1 + (-1 - 2\nu)t)\}$
- Val = $\{\}$

$$2. L_{JY} = z\tau^2 - (2 + 2\nu + 2x)\tau + z$$

- Constants: z, ν
- Solutions = $\{B_J(\nu + x, z), B_Y(\nu + x, z)\}$

- $\text{Gquo} = \{\frac{z}{4}t^2(1 + (-1 - 2\nu)t)\}$
 - $\text{Val} = \{\}$
3. $L_{WW} = \tau^2 + (z - 2\nu - 2x - 2)\tau - \nu - x - \frac{1}{4} - \nu^2 - 2\nu x - x^2 + \eta^2$
- Constants: z, ν, η
 - Solution = $\{W_W(\nu + x, \eta, z)\}$
 - $\text{Gquo} = \{(-3 + 2\sqrt{2})(1 + \frac{1}{2}\sqrt{2}zt), (-3 - 2\sqrt{2})(1 - \frac{1}{2}\sqrt{2}zt)\}$
 - $\text{Val} = \{[\eta + \frac{1}{2} - \nu, 1], [-\eta + \frac{1}{2} - \nu, 1]\}$ if $\eta \notin \frac{1}{2}\mathbb{Z}$ or $\text{Val} = \{[\eta + \frac{1}{2} - \nu, 2]\}$ if $\eta \in \frac{1}{2} + \mathbb{Z}$
4. $L_{WM} = (2\eta + 2\nu + 3 + 2x)\tau^2 + (2z - 4\nu - 4x - 4)\tau - 2\eta + 1 + 2\nu + 2x$
- Constants: z, ν, η
 - Solution = $\{W_M(\nu + x, \eta, z)\}$
 - $\text{Gquo} = \{1 - 2\sqrt{-zt}^{\frac{1}{2}} - 2zt, 1 + 2\sqrt{-zt}^{\frac{1}{2}} - 2zt\}$
 - $\text{Val} = \{[\eta + \frac{1}{2} - \nu, 1], [-\eta + \frac{1}{2} - \nu, 1]\}$ if $\eta \notin \frac{1}{2}\mathbb{Z}$ or $\text{Val} = \{[\eta + \frac{1}{2} - \nu, 2]\}$ if $\eta \in \frac{1}{2} + \mathbb{Z}$
5. $L_{2F_1} = (z - 1)(a + x + 1)\tau^2 + (-z + 2 - za - zx + 2a + 2x + zb - c)\tau - a + c - 1 - x$
- Constants: a, b, c, z
 - Solution = $\{2F_1(a + x, b; c; z)\}$
 - $\text{Gquo} = \{(1 - z)(1 + (c - 2b)t), \frac{1}{1-z}(1 + (2b - c)t)\}$
 - $\text{Val} = \{[-a, 1], [-a + c, 1]\}$ if $c \notin \mathbb{Z}$ or $\text{Val} = \{[-a, 2]\}$ if $c \in \mathbb{Z}$

In case 5, whenever $b \in [0, -1, -2, \dots]$, $2F_1(a + x, b; c; z)$ satisfies a first order recurrence equation as mentioned in [8, Remark 4.1]. So, this case is not of interest to this algorithm. Also, $u(x) = \frac{\Gamma(a+x+1-c)}{\Gamma(a+x)} 2F_1(a + x + 1 - c, b + 1 - c; 2 - c; z)$ is another solution of L_{2F_1} when $u(x)$ is defined and $c \notin \mathbb{Z}$ by [8, Theorem 4.4].

6. EXAMPLES

EXAMPLE 6.1. Sequence A096121 = [2, 8, 60, 816, 17520, 550080, ...] in [12] represents the “Number of full spectrum rook’s walks on a $(2 \times n)$ board” and it is a solution of the recurrence operator $A = \tau^2 - (x + 1)(x + 2)\tau - (x + 1)(x + 2)$. The local data of A is

$$\text{Gquo}(A) = \{-t^2(1 - t)\} \text{ and } \text{Val} = \{\}.$$

The local data of A matches the operator L_{IK} in the table in Section 5. Before we can call algorithm Find GT-transformation we need to find explicit values for the unknown constants z and ν appearing in L_{IK} . Since $\tau(B_J(\nu + x, z)) = B_J(\nu + x + 1, z)$ and τ is a gauge transformation, we only need $\nu \bmod \mathbb{Z}$. Comparing $\text{Gquo}(A)$ with $\text{Gquo}(L_{IK})$ (see table in Section 5) using Theorem 4.4 gives $-1 \equiv -1 - 2\nu \bmod \mathbb{Z}$ and $-\frac{z}{4} = -1$. Hence $\nu \in \frac{1}{2} + \mathbb{Z}$ or $\nu \in 0 + \mathbb{Z}$ and $z = \pm 2$. So, if A can be reduced to L_{IK} for some parameter value, then A can be reduced to one of:

$$-2\tau^2 + (2 + 2x)\tau + 2, \quad -2\tau^2 + (3 + 2x)\tau + 2,$$

$$2\tau^2 + (2 + 2x)\tau - 2, \quad 2\tau^2 + (3 + 2x)\tau - 2.$$

(These are L_{IK} with $\nu \in \{0, \frac{1}{2}\}$, $z \in \{2, -2\}$.) Then algorithm Find GT-transformation finds that A can be reduced to $-2\tau^2 + (2 + 2x)\tau + 2$. It also finds the gauge transformation 1 and the term product $\tau - (x + 1)$. From the list, a basis of solutions of $-2\tau^2 + (2 + 2x)\tau + 2$ is

$$\{B_I(x, -2), B_K(x, 2)\}.$$

By applying the gauge transformation and the term product we get a basis of solutions of A as

$$\{B_I(x, -2)\Gamma(x + 1), B_K(x, 2)\Gamma(x + 1)\}.$$

EXAMPLE 6.2. Sequence A005572 = [1, 4, 17, 76, 354, 1704, 8421, ...] in [12] represents the “Number of walks on cubic lattice starting and finishing on the xy -plane and never going below it” and it is a solution of the recurrence operator $H = (x + 4)\tau^2 + (-20 - 8x)\tau + (12x + 12)$. This same example has been used in [8] also. The local data of H is

$$\text{Gquo}(A) = \{\frac{1}{3}, 3\} \text{ and } \text{Val} = \{[0, 2]\}.$$

The local data of H matches with the operator L_{2F_1} in the table in Section 5. Since $\text{Val} = \{[0, 2]\}$, we get $a, c \in 0 + \mathbb{Z}$. We may take $a = 1$ and $c = 1$ so that $2F_1$ is defined. Comparing Gquo gives $c - 2b \equiv 0 \bmod \mathbb{Z}$ and $1 - z \in \{1/3, 3\}$. By Lemma 4.3 in [8] we need $b \bmod \mathbb{Z}$, so $b \in 0 + \mathbb{Z}$ or $b \in \frac{1}{2} + \mathbb{Z}$. So, if H can be reduced to L_{2F_1} for some parameter values, then H can be reduced to one of:

$$-3(2 + x)\tau^2 + (7 + 4x)\tau - 1 - x, \quad -3(2 + x)\tau^2 + (6 + 4x)\tau - 1 - x$$

$$-\frac{1}{3}(2 + x)\tau^2 + (\frac{5}{3} + \frac{4}{3}x)\tau - 1 - x, \quad -\frac{1}{3}(2 + x)\tau^2 + (2 + \frac{4}{3}x)\tau - 1 - x.$$

(These are L_{2F_1} with $a = 1, b \in \{0, \frac{1}{2}\}, c = 1, z \in \{-2, \frac{2}{3}\}$.) Then algorithm Find GT-transformation finds that H can be reduced to $-\frac{1}{3}(2 + x)\tau^2 + (2 + \frac{4}{3}x)\tau - 1 - x$ with gauge transformation $\frac{1}{x+2}(-\tau + 3)$ and term product $\tau - 2$. From the table, a solution of $-\frac{1}{3}(2 + x)\tau^2 + (2 + \frac{4}{3}x)\tau - 1 - x$ is

$$2F_1(x + 1, \frac{1}{2}; 1; \frac{2}{3}).$$

By applying the gauge transformation and the term product we get a solution of A and after checking initial values, we find that the sequence equals

$$\frac{2^{x+1}}{\sqrt{3}(x + 2)} \left(2F_1(x + 2, \frac{1}{2}; 1; \frac{2}{3}) \cdot 2 - 2F_1(x + 1, \frac{1}{2}; 1; \frac{2}{3}) \cdot 3 \right).$$

7. ALGORITHM

As in the examples in Section 6, our algorithm will compute a number of candidates (in Step 3), and then try to reduce the input equation to one of those candidates (in Step 4). Here is how the list of candidates given in Step (3h) were determined (the other cases in Step 3 can be done in the same way). Suppose we have an operator L that has $c_1(1 + d_2t) \in \text{Gquo}(L)$ and $\text{Val}(L) = \{[m, 2]\}$ where $c_1 \geq 1, d_2 \in \mathbb{C}$, and $m \in \mathbb{Z}$. Then L matches the operator $L_{2F_1} = (z - 1)(a + x + 1)\tau^2 + (-z + 2 - za - zx + 2a + 2x + zb - c)\tau - a + c - 1 - x$ in Section 5. If L can be reduced to an operator L_{2F_1} then c_1 should equal either $1 - z$ or $\frac{1}{1-z}$. From $\text{Val}(L)$ we get $a = -m$ and $c \in \mathbb{Z}$. Since we

need $c \bmod \mathbb{Z}$ [8, Lemma 4.3], we may let $c = 0$. Also, we need $c - 2b$ or $2b - c \bmod \mathbb{Z}$, so candidates for b are $-\frac{d_2}{2}$ and $-\frac{d_2+1}{2}$. Hence candidates for the parameters $[a, b, c, z]$ are $[-m, -\frac{d_2}{2}, 0, 1 - c_1]$, $[-m, -\frac{d_2+1}{2}, 0, 1 - c_1]$, $[-m, -\frac{d_2}{2}, 0, 1 - \frac{1}{c_1}]$, $[-m, -\frac{d_2+1}{2}, 0, 1 - \frac{1}{c_1}]$. The other cases in Step 3 of the following algorithm were generated similarly.

Algorithm: solver

Input: An operator $L = a_2\tau^2 + a_1\tau + a_0 \in \mathbb{Q}(x)[\tau]$.

Output: At least one solution of L if there is an operator in the table in Section 5 to which L can be reduced. Otherwise the empty set.

1. $comb := \emptyset, const := \emptyset$
2. Calculate $Gquo(L)$ and $Val(L)$.
3. For $f \in Gquo(L)$ let $f = c_1t^v(1 + d_1t^{1/2} + d_2t)$.
 - (a) If $v = 2, c_1 > 0, d_1 = 0$, and $Val = \{\}$ then
 - i. Let $Z := \{2\sqrt{c_1}, -2\sqrt{c_1}\}, V := \{-\frac{1}{2}d_2 - \frac{1}{2}, -\frac{1}{2}(d_2 + 1) - \frac{1}{2}\}$
 - ii. $comb := \{z\tau^2 - (2 + 2v + 2x)\tau + z \mid v \in V, z \in Z\}$
 - (b) If $v = 2, c_1 < 0, d_1 = 0$, and $Val = \{\}$ then
 - i. Let $Z := \{2\sqrt{-c_1}, -2\sqrt{-c_1}\}, V := \{-\frac{1}{2}d_2 - \frac{1}{2}, -\frac{1}{2}(d_2 + 1) - \frac{1}{2}\}$
 - ii. $comb := \{z\tau^2 - (2 + 2v + 2x)\tau + z \mid v \in V, z \in Z\}$
 - (c) If $v = 0, d_1 = 0, Val(L) = \{[m, 2]\}$, and $c_1 = -3 + 2\sqrt{2}$ then
 - i. Let $NV := \{[0, \frac{1}{2} - m], [\frac{1}{2}, -m]\}, z := \text{rational part of } \sqrt{2}d$
 - ii. $comb := \{\tau^2 + (z - 2\nu - 2x - 2)\tau - \nu - x - \frac{1}{4} - \nu^2 - 2\nu x - x^2 + n^2 \mid [n, \nu] \in NV\}$
 - (d) If $v = 0, d_1 = 0, Val(L) = \{[n_1, 1], [n_2, 1]\}$, and $c_1 = -3 + 2\sqrt{2}$ then
 - i. Let $NV := \{[-\frac{n_1+n_2}{2}, \frac{n_1-n_2-1}{2}], [-\frac{n_1+n_2}{2}, -\frac{n_1+n_2-1}{2}], [-\frac{n_1+n_2+1}{2}, \frac{n_1-n_2}{2}], [-\frac{n_1+n_2+1}{2}, -\frac{n_1+n_2}{2}]\}, z := \text{rational part of } \sqrt{2}d$
 - ii. $comb := \{\tau^2 + (z - 2\nu - 2x - 2)\tau - \nu - x - \frac{1}{4} - \nu^2 - 2\nu x - x^2 + n^2 \mid [n, \nu] \in NV\}$
 - (e) If $v = 0, d_1 \neq 0, Val(L) = \{[m, 2]\}$, and $c_1 = 1$ then
 - i. Let $NV := \{[0, \frac{1}{2} - m], [\frac{1}{2}, -m]\}, z := -\frac{d_1^2}{4}$
 - ii. $comb := \{\tau^2(2n + 2\nu + 3 + 2x) + (2z - 4\nu - 4x - 4)\tau - 2n + 1 + 2\nu + 2x \mid [n, \nu] \in NV\}$
 - (f) If $v = 0, d_1 = 0, Val(L) = \{[n_1, 1], [n_2, 1]\}$, and $c_1 = -3 + 2\sqrt{2}$ then
 - i. Let $NV := \{[-\frac{n_1+n_2}{2}, \frac{n_1-n_2-1}{2}], [-\frac{n_1+n_2}{2}, -\frac{n_1+n_2-1}{2}], [-\frac{n_1+n_2+1}{2}, \frac{n_1-n_2}{2}], [-\frac{n_1+n_2+1}{2}, -\frac{n_1+n_2}{2}]\}, z := -\frac{d_1^2}{4}$
 - ii. $comb := \{\tau^2(2n + 2\nu + 3 + 2x) + (2z - 4\nu - 4x - 4)\tau - 2n + 1 + 2\nu + 2x \mid [n, \nu] \in NV\}$
 - (g) If $v = 0, d_1 = 0, c_1 \geq 1$, and $Val(L) = \{[m, 2]\}$ then

- i. Let $const := \{[-m, -\frac{d_2}{2}, 0, 1 - c_1], [-m, -\frac{d_2+1}{2}, 0, 1 - c_1], [-m, -\frac{d_2}{2}, 0, 1 - \frac{1}{c_1}], [-m, -\frac{d_2+1}{2}, 0, 1 - \frac{1}{c_1}]\}$
 - ii. $comb := \{(z - 1)(a + x + 1)\tau^2 + (-z + 2 - za - zx + 2a + 2x + zb - c)\tau - a + c - 1 - x \mid [a, b, c, z] \in const\}$
- (h) If $v = 0, d_1 = 0, c_1 \geq 1$, and $Val(L) = \{[n_1, 1], [n_2, 1]\}$ then
- i. Let $const := \{[-n_1, -\frac{n_2-n_1-d_2}{2}, n_2 - n_1, 1 - c_1], [-n_1, -\frac{n_2-n_1-d_2+1}{2}, n_2 - n_1, 1 - c_1], [-n_1, -\frac{n_2-n_1-d_2}{2}, n_2 - n_1, 1 - \frac{1}{c_1}], [-n_1, -\frac{n_2-n_1-d_2+1}{2}, n_2 - n_1, 1 - \frac{1}{c_1}], [-n_2, -\frac{n_1-n_2-d_2}{2}, n_1 - n_2, 1 - c_1], [-n_2, -\frac{n_1-n_2-d_2+1}{2}, n_1 - n_2, 1 - c_1], [-n_2, -\frac{n_1-n_2-d_2}{2}, n_1 - n_2, 1 - \frac{1}{c_1}], [-n_2, -\frac{n_1-n_2-d_2+1}{2}, n_1 - n_2, 1 - \frac{1}{c_1}]\}$
 - ii. For each $[a, b, c, z] \in const$, if ${}_2F_1(a, b; c; z)$ is not defined, then shift a, b, c by a suitable integer (this changes ${}_2F_1$ only by a gauge transformation [8, Lemma 4.3]).
 - iii. $comb := \{(z - 1)(a + x + 1)\tau^2 + (-z + 2 - za - zx + 2a + 2x + zb - c)\tau - a + c - 1 - x \mid [a, b, c, z] \in const\}$
- (i) Otherwise stop and return $\emptyset$.
4. For each $L_c \in comb$ check if L can be reduced to L_c and if so
- (a) Generate a basis of solutions or a solution of L_c by plugging in corresponding parameters.
 - (b) Apply the term transformation and gauge transformation to the result from Step (4a).
 - (c) Return the result of Step (4b) as output and stop the algorithm.

8. IMPROVEMENTS

From [12], we can obtain a large number of equations for which useful things are known, such as references, formulas, etc. These equations can be added to the table. Since they do not contain parameters, the key problem solved in this paper (finding parameter values) becomes empty, and is replaced with a new problem: how to quickly select the right equation from a large collection? This problem was solved in G. Levy's Ph.D thesis [8] using the p -curvature. Also treated in [8] was the reduction to L_{2F_1} , and to $\tau^2 + b_0$ (Liouvillian solutions). The thesis and implementation can be copied from [8].

8.1 Future work

The table in Section 5 contains a small number of base equations. We want to extend this table significantly. Given an equation and solution(s), it is easy to add the equation to the table; all we have to do is to compute its local data with our implementation and compute formulas for the parameters from that. We will need to set up a systematic approach to construct base equations and their solutions to ensure that none will be overlooked. Also for higher order, we can apply the same techniques.

9. APPENDIX

In this section, we will discuss more about indicial equation of a linear difference operator and state Lemma 9.3, which has been used to prove Theorem 4.4.

Let $u \in K$, $u \neq 0$, and $v(u) = n$. Write

$$u = c_n t^n + c_{n+1} t^{n+1} + \dots$$

then it follows from equation (2) (use the fact that $L(a+b) = L(a) + L(b)$) that

$$L(u) = c_n P(n) t^{n+v(L)} + \dots$$

COROLLARY 9.1. *Let $u \in K$, $u \neq 0$, then*

$$v(L(u)) = v(u) + v(L) \iff v(u) \text{ is not a root of } \text{Ind}_L$$

LEMMA 9.2. *Let $L \in K[\tau]$ and $L \neq 0$. Then*

$$\dim(\text{Ker}(L, K)) > 0 \iff \text{mult}_{\mathbb{Z}}(\text{Ind}_L) > 0$$

where Ind_L is the indicial equation of L , and $\text{mult}_{\mathbb{Z}}(\text{Ind}_L)$ denotes the number of integer roots of Ind_L .

PROOF. " $\implies$ " if $u \in K$, $u \neq 0$, and $L(u) = 0$ then $v(u)$ must be a root of Ind_L by Corollary 9.1.

" $\impliedby$ " Let n be the largest integer root of Ind_L , so

$$\text{Ind}_L(n) = 0, \text{Ind}_L(n+1) \neq 0, \text{Ind}_L(n+2) \neq 0, \dots \quad (4)$$

Since $\text{Ind}_L(n) = 0$ it follows from equation (2) that

$$L(t^n) = t^{n+v(L)} \cdot (0t^0 + a_1 t^1 + a_2 t^2 + \dots)$$

Write

$$u = t^n + c_{n+1} t^{n+1} + c_{n+2} t^{n+2} + \dots$$

Then write

$$L(u) = t^{n+v(L)} \cdot (0t^0 + A_1 t^1 + A_2 t^2 + \dots)$$

Now $A_1 = a_1 + c_{n+1} \text{Ind}_L(n+1)$ and since $\text{Ind}_L(n+1) \neq 0$ there is a unique $c_{n+1} \in \mathbb{C}$ for which A_1 vanishes, namely $c_{n+1} := -a_1 / \text{Ind}_L(n+1)$. Then A_2 equals some constant plus $c_{n+2} \text{Ind}_L(n+2)$, and again $\text{Ind}_L(n+2) \neq 0$ so there is a unique c_{n+2} for which A_2 vanishes. Continuing this way leads to $L(u) = 0$. $\square$

We can say the same for $L \in K_r[\tau]$ for $r \in \mathbb{N}$.

LEMMA 9.3. *Let $L \in K[\tau]$ and $L \neq 0$. Then*

$$\dim(\text{Ker}(L, K_r)) > 0 \iff \text{mult}_{\frac{1}{r}\mathbb{Z}}(\text{Ind}_L) > 0$$

where Ind_L is the indicial equation of L , and $\text{mult}_{\frac{1}{r}\mathbb{Z}}(\text{Ind}_L)$ denotes the number of roots of Ind_L in $\frac{1}{r}\mathbb{Z}$.

10. REFERENCES

- [1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, 1964, New York, ninth Dover printing, tenth GPO printing, ISBN 0-486-61272-4
- [2] S.A. Abramov, M. A. Barkatou, M. van Hoeij, M. Petkovšek, *Subanalytic Solutions of Linear Difference Equations and Multidimensional Hypergeometric Sequences*, to appear in J. Symbolic Comput.
- [3] M. A. Barkatou, *Rational Solutions of Matrix Difference Equations: The Problem of Equivalence and Factorization*, ISSAC'1999, p. 277–282, (1999).
- [4] M. A. Barkatou and G. Chen, *Computing the exponential part of a formal fundamental matrix solution of a linear difference system*, J. Diff. Equ. Appl. **5**, p. 117–142, (1999)
- [5] G. D. Birkhoff, *Formal theory of irregular linear difference equations*, Acta Math., **54**, p. 205–246, (1930).
- [6] T. Cluzeau, M. van Hoeij, *Computing hypergeometric solutions of linear difference equations*, AAEECC, **17(2)**, p. 83–115, (2006).
- [7] Yongjae Cha and Mark van Hoeij, *Liouvillian solutions of irreducible linear difference equations*, ISSAC '09: Proceedings of the 2009 international symposium on Symbolic and algebraic computation, p.87–94,(2009).
- [8] G. Levy, *Solutions of second order recurrence relations*, Ph.D. dissertation, Florida State University, (2010). Thesis and implementation available at www.math.fsu.edu/~glevy
- [9] M. van Hoeij, *Finite singularities and hypergeometric solutions of linear recurrence equations*, J. Pure Appl. Algebra, **139**, p. 109–131, (1999).
- [10] P. A. Hendriks, M. F. Singer, *Solving difference equations in finite terms*, J. Symbolic Comput., **27**, p. 239–259, (1999).
- [11] A. H. M. Levelt, A. Fahim, *Characteristic Classes for Difference Operators*, Compos. Math, **117**, p. 223–241, (1999).
- [12] N. J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences*, www.research.att.com/~njas/sequences (2009)
- [13] M. Petkovšek, H. S. Wilf, D. Zeilberger, *A = B*, With a foreword by Donald E. Knuth. A. K. Peters, Ltd., Wellesley, MA, (1996).
- [14] M. van der Put, M. F. Singer, *Galois Theory of Difference Equations*, Springer-Verlag, **1666**, Lecture Notes in Mathematics, (1997).

On Some Decidable and Undecidable Problems Related to Q-Difference Equations with Parameters*

S. A. Abramov

Computing Centre of the Russian Academy of Sciences,
ul. Vavilova 40, Moscow, 119991
Russia
sergeyabramov@mail.ru

ABSTRACT

We consider linear q -difference equations with polynomial coefficients depending on parameters. For the case when the ground field is $\mathbb{Q}(q)$ we propose an algorithm recognizing whether or not there exist numerical values of parameters for which a given equation has a non-zero polynomial solution (alternatively, a rational-function solution). We prove that there exists no such algorithm if the parameter values are polynomials or rational functions of q .

Categories and Subject Descriptors

I.1.2 [Symbolic And Algebraic Manipulation]: Algorithms—*Algebraic algorithms*

General Terms

Algorithms, Theory

Keywords

q -difference equations with parameters, polynomial solutions, rational-function solutions, undecidable problems

1. INTRODUCTION

Suppose that in an equation $L(y) = 0$ the operator L is of the form

$$r_\rho(x, t_1, \dots, t_m)D^\rho + r_{\rho-1}(x, t_1, \dots, t_m)D^{\rho-1} + \dots \quad (1) \\ \dots + r_0(x, t_1, \dots, t_m),$$

where $D = \frac{d}{dx}$, and $r_0, r_1, \dots, r_\rho$ are polynomials over $\mathbb{Q}$ in the specified variables, and $t_1, t_2, \dots, t_m$ are parameters. In the paper [13] of D. Boucher the following result of J.-A. Weil is mentioned: there is no algorithm that, for an arbitrary operator L of form (1) answers whether or not numerical values of parameters $t_1, t_2, \dots, t_m$ exist for which equation $L(y) = 0$

*This work was supported in part by the Russian Foundation for Basic Research, project no. 10-01-00249, and by ECONET, project no. 21315ZF.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

has a solution in the form of a non-zero rational function of x . The proof is based on the David-Matiyasevich-Putnam-Robinson theorem which says that there exists no algorithm which, for an arbitrary polynomial $P(t_1, t_2, \dots, t_m)$ with integral coefficients, determines whether or not the equation $P(t_1, t_2, \dots, t_m) = 0$ has an integral solution [19]. The result by Weil can be easily extended to the problem of existence of polynomial solutions of equation $L(y) = 0$.

Similar results have been obtained for the difference case ([2, 3]). The operator L is of the form

$$r_\rho(x, t_1, \dots, t_m)E^\rho + r_{\rho-1}(x, t_1, \dots, t_m)E^{\rho-1} + \dots \quad (2) \\ \dots + r_0(x, t_1, \dots, t_m),$$

where E is the shift operator: $E(y(x)) = y(x+1)$, and again $r_0, r_1, \dots, r_\rho$ are polynomials over $\mathbb{Q}$ in the specified variables, $t_1, t_2, \dots, t_m$ are parameters.

In this paper we consider q -difference equations. Differential equations are based on the differentiation operator D , while difference equations are based on the shift operator E . In turn, the q -difference equations are based on the q -shift operator Q :

$$Q(y(x)) = y(qx),$$

where q is a fixed value or an additional variable (q -calculus and the theory and algorithms for q -difference equations are of interest in combinatorics, especially in the theory of partitions [10, Sect. 8.4], [11]). The q -difference analogue of operators (1), (2) is

$$r_\rho(x, t_1, \dots, t_m)Q^\rho + r_{\rho-1}(x, t_1, \dots, t_m)Q^{\rho-1} + \dots \quad (3) \\ \dots + r_0(x, t_1, \dots, t_m),$$

where $r_0, r_1, \dots, r_\rho$ are polynomials in specified variables over a field k of characteristic 0. We assume that $k = k_0(q)$, where k_0 is a subfield of k , and q, x are algebraically independent over k_0 .

We show that the situation with the parametric case for q -difference equations in some sense is more interesting than for differential and difference equations. Let, e.g., the ground field k be $\mathbb{Q}(q)$. Then there is an algorithm that recognizes the existence of numerical (real, complex) values of the parameters for which a given linear q -difference equation has a solution in the form of a non-zero polynomial or, alternatively, rational function; it is possible that the right-hand side is a non-zero polynomial in x that contains parameters. (Recall that a rational solution of a linear q -difference equation with polynomial coefficients and polynomial right-hand side without parameters is a rational function of x over k such that substituting it into the equation gives an equality in $k(x)$.)

At the same time, if the values of parameters are allowed to be arbitrary polynomials or rational functions of q then such algorithm does not exist.

Acknowledgments. I am grateful to M. Petkovšek and J.-A. Weil for interesting discussions, and to T. Pheidas and A. Shen for valuable consultations. I also express my thanks to D. Khmelnov, A. Ryabenko, and anonymous referees for their helpful remarks.

2. PRELIMINARIES: Q -DIFFERENCE EQUATIONS WITHOUT PARAMETERS AND SYSTEMS OF ALGEBRAIC EQUATIONS

In this section we consider linear q -difference equations with polynomial coefficients and polynomial right-hand sides which do not contain any parameters, i.e., equations of the form

$$L(y) = f(x), \quad (4)$$

where

$$L = r_\rho(x)Q^\rho + \dots + r_1(x)Q + r_0(x), \quad (5)$$

$r_0(x), r_1(x), \dots, r_\rho(x), f(x) \in k[x]$. Here k is a field of characteristic 0, $k = k_0(q)$, where k_0 is a subfield of k , and q , x are algebraically independent over k_0 . We will assume that $r_\rho(x), r_0(x) \in k[x] \setminus \{0\}$, and ρ will be called the *order* of L (we write $\text{ord } L = \rho$). Below we describe briefly some known algorithms for computing all polynomial and rational-function solutions of equations of this form (see [1], [4]; an implementation of some versions of these algorithms is available in the standard package QDifferenceEquations of Maple computer algebra system [23]). We will need these algorithms later when we consider equations with parameters. An observation given in [12] will be also valuable for us.

In addition we discuss some facts related to systems of algebraic equations.

2.1 An algorithm for finding polynomial solutions

We connect with operator (5) the non-negative integer number ω and the polynomial $I(\lambda) \in k[\lambda]$:

$$\omega = \max_{0 \leq j \leq \rho} \deg r_j(x), \quad I(\lambda) = \sum_{\substack{0 \leq j \leq \rho \\ \deg r_j(x) = \omega}} \text{lc}(r_j(x)) \lambda^j \quad (6)$$

($\text{lc}(\dots)$ is the leading coefficient of a polynomial belonging to $k[x] \setminus \{0\}$). The algebraic equation $I(\lambda) = 0$ is called the *indicial equation*, and the integer ω is called the *increment* connected with operator L . Set the degree of zero polynomial to be $-\infty$. The following statement demonstrates the role of the indicial equation in the search for polynomial solutions.

Let $\varphi(x)$ be a polynomial solution of the equation $L(y) = f(x)$, $f(x) \in k[x]$. Then $\deg \varphi(x)$ does not exceed

$$l = \max\{\deg f - \omega, \tilde{\lambda}\}, \quad (7)$$

where $\tilde{\lambda} = \max\{\{h \in \mathbb{N} : I(q^h) = 0\} \cup \{-\infty\}\}$.

The statement is justified by the fact that if $\varphi(x) \in k[x]$, $\deg \varphi(x) = d$, $I(q^d) \neq 0$ then $\deg L(\varphi(x)) = d + \omega$.

There is an algorithm which allows for any algebraic equation with one unknown λ over the field $k = k_0(q)$ to find all roots of the form q^h , $h \in \mathbb{Z}$: since $q^h \neq 0$ for any $h \in \mathbb{Z}$, we can assume that the algebraic equation has the form

$$a_s(q)\lambda^s + \dots + a_1(q)\lambda + a_0(q) = 0 \quad (8)$$

$a_1(q), a_2(q), \dots, a_{s-1}(q) \in k_0[q]$, $a_0(q), a_s(q) \in k_0[q] \setminus \{0\}$. If q^h is a root, then $q^h | a_0(q)$ when $h \geq 0$, and $q^{-h} | a_s(q)$ when $h < 0$.

The simplest version of the algorithm for finding the general polynomial solution is to find an upper bound for degrees of all possible polynomial solutions and to use the undetermined coefficients method. A faster algorithm is described in [6].

2.2 An algorithm for finding rational-function solutions

The general principle of the search for rational solutions that we use is as follows: first of all to find a *universal factor* which is a rational function $V(x)$ over k such that if the original equation has a rational solution, then this solution can be written in the form $z(x)V(x)$, where $z(x)$ is a polynomial. Of course, it is possible that $z(x) \nmid \text{den } V(x)$ (we write $a(x) \nmid b(x)$, if polynomials $a(x), b(x) \in k[x]$ have a common factor of positive degree). The substitution

$$y(x) = z(x)V(x) \quad (9)$$

into the original equation reduces the problem of finding rational-function solutions to the problem of finding polynomial solutions.

We describe an algorithm for finding a universal factor.

Any polynomial $\varphi(x) \in k[x] \setminus \{0\}$ can be represented in the form $\varphi(x) = x^v b(x)$, where $v \in \mathbb{N}$ and the polynomial $b(x)$ is not divisible by x . If $\varphi(x)$ is the zero polynomial, then set $v = \infty$. We denote v by $\nu(\varphi(x))$ and (as usual) call it the *valuation* of $\varphi(x)$. If $\nu(\varphi(x)) = \nu(\psi(x)) = 0$ for $\varphi(x), \psi(x) \in k[x]$, then we can consider the *q -dispersion set* (finite) of polynomials $\varphi(x)$ and $\psi(x)$:

$$\text{qds}(\varphi(x), \psi(x)) = \{h \in \mathbb{N} : \varphi(x) \nmid \psi(q^h x)\} \quad (10)$$

and their *q -dispersion*:

$$\text{qdis}(\varphi(x), \psi(x)) = \max(\text{qds}(\varphi(x), \psi(x)) \cup \{-\infty\}). \quad (11)$$

The set $\text{qds}(\varphi(x), \psi(x))$ can be found, e.g., by computing all the roots having the form $\lambda = q^h$, $h \in \mathbb{N}$, of the equation $R(\lambda) = 0$, where $R(\lambda) = \text{Res}_x(\varphi(x), \psi(\lambda x))$. (In [5] an algorithm is proposed which works also in the case when q is an algebraic number which is not a root of unity.)

A universal factor can be found in the form

$$V(x) = x^{l_0} \cdot \frac{1}{U(x)}, \quad (12)$$

where $l_0 \in \mathbb{Z}$, $U(x) \in k[x]$, $\nu(U(x)) = 0$. The polynomial $U(x)$ can be constructed by the following algorithm ([1], [4]):

Set $A(x) = \tilde{r}_\rho(q^{-\rho}x)$, $B(x) = \tilde{r}_0(x)$, where $\tilde{r}_\rho(x) = \frac{r_\rho(x)}{x^{\nu(r_\rho(x))}}$, $\tilde{r}_0(x) = \frac{r_0(x)}{x^{\nu(r_0(x))}}$. Compute $H = \text{qds}(A(x), B(x))$. If $H = \emptyset$, then stop the algorithm with the result $U(x) = 1$ (we assume in the rest of this description of the algorithm that $H = \{h_1, h_2, \dots, h_s\}$, $h_1 > h_2 > \dots > h_s$, $s \geq 1$). Set $U(x) = 1$ and for all h_i , starting from h_1 in the decreasing

order, execute the following assignments:

$$\begin{aligned} N(x) &= \gcd(A(x), B(q^{h_i}x)) \\ A(x) &= A(x)/N(x) \\ B(x) &= B(x)/N(q^{-h_i}x) \\ U(x) &= U(x) \prod_{j=0}^{h_i} N(q^{-j}x). \end{aligned}$$

The final value of $U(x)$ is a polynomial which can be used to construct a universal factor (12).

For finding l_0 we assign to L of the form (5) and to the equation $L(y) = f(x)$, $f(x) \in k[x]$, the increment

$$\omega_0 = \min_{0 \leq j \leq \rho} \nu(r_j(x)) \quad (13)$$

and the indicial equation $I_0(\lambda) = 0$, where

$$I_0(\lambda) = \sum_{\substack{0 \leq j \leq \rho \\ \nu(r_j(x)) = \omega_0}} \text{tc}(r_j(x)) \lambda^j \quad (14)$$

($\text{tc}(\dots)$ is the trailing coefficient of a polynomial belonging to $k[x] \setminus \{0\}$). We can set

$$l_0 = \min\{\nu(f(x)) - \omega_0, \tilde{\lambda}_0\}, \quad (15)$$

where

$$\tilde{\lambda}_0 = \min(\{h \in \mathbb{Z} : I_0(q^h) = 0\} \cup \{\infty\}). \quad (16)$$

This can be justified by the fact that if $F(x) \in k(x)$, and $\hat{F}(x)$ is a formal Laurent series in x for $F(x)$, $n = \nu(\hat{F}(x))$ is the valuation of this series (i.e., the minimal exponent of x in non-zero terms; for zero series the valuation is ∞) and $I_0(h^n) \neq 0$, then $\nu(L(\hat{F}(x))) = n + \omega_0$. Notice that in the algorithm from [1], [4] the value l_0 is computed in a different way.

REMARK 1. Let $A(x), B(x)$ be as in the algorithm description given above, and $U(x)$ be the result of applying this algorithm. Let $d \geq \text{qdis}(A(x), B(x))$. Using the same reasonings as in [12] for the difference case, one can show that $U(x)$ divides the polynomial $\prod_{i=0}^d r_\rho(q^{-\rho-i}x)$. This implies that the latter polynomial can be used in (12) instead of $U(x)$.

REMARK 2. The existence of the roots having the form q^h , $h \in \mathbb{Z}$, of the equation $I_0(\lambda) = 0$ is a necessary condition for the existence of non-zero rational-function (in particular, polynomial) solutions of $L(y) = 0$.

Another algorithm for finding a universal factor was described in [18] where difference equations were discussed, but it was noted that the proposed approach can be used in the q -difference case as well. However for the purposes of this paper the algorithm described above (especially in the form mentioned in Remark 1) is more suitable.

2.3 Pairs of systems of algebraic equations

Working with parameters we will face systems of algebraic equations (nonlinear in general). A well-known problem is recognizing whether or not a given system with coefficients in a field k_0 has a solution whose components belong to an extension Λ of k_0 . We will consider also a more general problem: given a pair (S_1, S_2) of systems of algebraic equations (possibly empty), decide whether there are values of the unknowns belonging to Λ which satisfy all equations in S_1 , but – provided that $S_2 \neq \emptyset$ – not all equations in S_2 . (If $S_1 = \emptyset$, then by definition any set of values of the unknowns satisfies

S_1 .) We will refer to this more general problem as *problem $\mathcal{S}_{k_0, \Lambda}$* .

If the problem $\mathcal{S}_{k_0, \Lambda}$ is decidable for given k_0, Λ then we will denote by $A_{k_0, \Lambda}$ an algorithm which solves this problem. The result of applying $A_{k_0, \Lambda}$ to a pair of systems is one of the words “yes”, “no”.

If $k_0 = \mathbb{Q}$, then $\mathcal{S}_{k_0, \Lambda}$ is decidable for all Λ from the list

$$\mathbb{C}, \mathbb{R}, \overline{\mathbb{Q}}, \mathbb{R} \cap \overline{\mathbb{Q}}. \quad (17)$$

Using the Groebner bases technique an algorithm for $\mathbb{C}$ and $\overline{\mathbb{Q}}$ as Λ can be obtained ([14, Sect. 6], [17, Sect. 21.6], [20, Ch. 4], etc.), and using Tarski’s theorem one can obtain an algorithm for $\mathbb{R}$ and $\mathbb{R} \cap \overline{\mathbb{Q}}$ as Λ ([16], [20, Sect. 8.6.3]). It is also known for the case $k_0 = \mathbb{Q}$ that a solution with components in $\Lambda = \mathbb{C}$ exists iff there exists a solution with components in $\Lambda = \overline{\mathbb{Q}}$, while a solution with components in $\Lambda = \mathbb{R}$ exists iff there exists a solution with components in $\Lambda = \mathbb{R} \cap \overline{\mathbb{Q}}$.

If for an arbitrary equation in one variable with coefficients in k_0 we can recognize the existence of a root in Λ , then in the case of one variable an algorithm $A_{k_0, \Lambda}$ can be based on the Euclidean algorithm (see Section 4.2.4).

3. BOUNDING INTEGER EXPONENTS OF ROOTS OF ALGEBRAIC EQUATIONS

PROPOSITION 1. Let there exist at least one non-zero element among $b_0(q), b_1(q), \dots, b_u(q) \in k_0[q]$. Then the inequality

$$|h| \leq \max_{0 \leq j \leq u} \deg_q b_j(q) \quad (18)$$

is valid for all $h \in \mathbb{Z}$ such that q^h is a root of the equation

$$b_u(q)\lambda^u + \dots + b_1(q)\lambda + b_0(q) = 0. \quad (19)$$

Proof. See the algorithm for finding the roots of the form q^h , $h \in \mathbb{Z}$, in Section 2.1. $\square$

We will show that the computation of the roots q^h , $h \in \mathbb{Z}$, in algorithms of Sections 2.1, 2.2 for finding polynomial and rational-function solutions can be replaced by finding an upper bound of $|h|$. In Section 4 this will be used for q -difference equations with parameters, but in the current section we still consider equation (4) that does not have any parameters. We can clear denominators in coefficients and $f(x)$ (those denominators are polynomials in q), and assume that $r_0(x), r_1(x), \dots, r_\rho(x), f(x) \in k_0[q][x]$ in (4), (5). It will be convenient for us in some situations to consider coefficients and right-hand sides of q -difference equations as polynomials in q and x over k_0 . However we will use as a rule the notation $r_0(x), r_1(x), \dots, r_\rho(x), f(x)$ etc, because the variable x is the main one: we produce the q -shift w.r.t. x . (In some cases we will write just $r_0, r_1, \dots, r_\rho, f$.) When we write, e.g., $\text{lc}(f)$, then we have in mind the leading coefficient of f as a polynomial in x , and this leading coefficient is a polynomial in q over k_0 ; the same goes for the trailing coefficient $\text{tc}(f)$. However we will use $\deg_x$ resp. $\deg_q$ for degrees of polynomials in x resp. q .

Notice that $\text{lc}(r_j)$ in $I(\lambda)$ (see (6)) and $\text{tc}(r_j)$ in $I_0(\lambda)$ (see (14)) are polynomials in q of degree $\leq w_q$.

PROPOSITION 2. Let the coefficients of operator (5) belong to $k_0[q, x]$, and w_q resp. w_x be maximal degrees in q resp. x of all these coefficients. Let $f \in k_0[q, x]$. Then

(i) the degree of any polynomial solution of $L(y) = f$ does not exceed

$$\max\{\deg_x f, w_q\}, \quad (20)$$

(ii) any rational-function solution of $L(y) = f$ can be represented as the product of a polynomial and the rational function

$$V(x) = \frac{1}{x^w \prod_{i=0}^d r_\rho(q^{-\rho-i}x)}, \quad (21)$$

where $w = \max\{w_x, w_q\}$, $d = \rho w_x^2 + 2w_x w_q$.

Proof. (i) The value (20) cannot be less than (7).

(ii) Going back to the algorithm for computing $U(x)$ given in Section 2.2, set $A'(x) = q^{\rho w_x} A(x)$. We have $\text{qdis}(A(x), B(x)) = \text{qdis}(A'(x), B(x))$, and $A'(x), B(x)$ can be considered as polynomials in q and x over k_0 . Then

$$\begin{aligned} \deg_x A' &\leq w_x, & \deg_x B &\leq w_x, \\ \deg_q A' &\leq w_q + \rho w_x, & \deg_q B &\leq w_q. \end{aligned}$$

Taking into account the form of the Sylvester matrix of polynomials $A'(x)$, $B(\lambda x)$ and the algorithm for computing the q -dispersion using a resultant, we get

$$\begin{aligned} \deg_q \text{Res}_x(A'(x), B(\lambda x)) &\leq \\ &\leq \deg_q A' \deg_x B + \deg_q B \deg_x A' \\ &\leq (w_q + \rho w_x)w_x + w_q w_x. \end{aligned}$$

This and Proposition 1 imply $\text{qdis}(A(x), B(x)) \leq \rho w_x^2 + 2w_x w_q = d$. So (ii) follows from Remark 1 and from inequalities $-\omega_0 \geq -w_x$, $\lambda_0 \geq -w_q$ (therefore $w \geq -l_0$ for l_0 , computed by formula (15)). $\square$

4. Q -DIFFERENCE EQUATIONS WITH PARAMETERS, INDEPENDENT OF Q

We will show that the algorithmic problems mentioned in the Introduction, undecidable in the differential and difference cases, are decidable in the q -difference case when parameters are independent of q .

Computation of roots will be replaced by finding some bounds for the exponents h (see Section 3). Of course, using the bounds instead of exact values of the exponents increases performance time of the algorithms. But, first, concerning q -difference equations with parameters, the problem of finding such exact values appears to be unsolvable. Second, we will be interested only in establishing the existence of algorithms. The effectiveness questions will not be considered (the only exception is Section 4.2.4).

4.1 Basic assumptions

Here we formulate some assumptions which will remain valid throughout Section 4.

1. Λ is an extension of the field k_0 of characteristic 0, and q, x are algebraically independent over Λ .
2. The algorithmic problem $\mathcal{S}_{k_0, \Lambda}$ is decidable, i.e., there exists an algorithm $A_{k_0, \Lambda}$ (see Section 2.3).
3. The operator L has the form

$$r_\rho Q^\rho + r_{\rho-1} Q^{\rho-1} + \dots + r_0, \quad (22)$$

where $r_0, r_1, \dots, r_\rho \in k_0[q, x, t_1, t_2, \dots, t_m]$ and $t_1, t_2, \dots, t_m$ are parameters. The right-hand side f of the equation $L(y) = f$ also belongs to $k_0[q, x, t_1, t_2, \dots, t_m]$.

4.2 Recognizing existence of polynomial and rational-function solutions in the homogeneous case

Till Section 4.3 we assume that a given q -difference equation with parameters is homogeneous, i.e., of the form $L(y) = 0$.

First we consider the question of existence of $\tau_1, \tau_2, \dots, \tau_m \in \Lambda$ such that the equation $L(y) = 0$ after substituting $\tau_1, \tau_2, \dots, \tau_m$ for $t_1, t_2, \dots, t_m$ becomes an equation with a non-zero solution in $\Lambda[q, x]$ resp. in $\Lambda(q, x)$ (but notice that the unknown function is denoted by $y(x)$, not by $y(q, x)$). We will refer to the two algorithmic problems related to the existence of parameter values such that the corresponding equation has non-zero polynomial resp. rational-function solutions, as *problem* $\mathcal{P}_{k_0, \Lambda}$ resp. *problem* $\mathcal{R}_{k_0, \Lambda}$. We will show in particular that if $k_0 = \mathbb{Q}$, then both problems are decidable when Λ is any field from the list (17).

Any parameter values belonging to Λ such that a given q -difference equation has a non-zero polynomial resp. rational-function solution will be called *adequate*.

Now we introduce a notion which will be useful in the sequel. Let $\varphi \in k_0[q, x, t_1, t_2, \dots, t_m]$. The system of algebraic equations in $t_1, t_2, \dots, t_m$, which is produced by representing φ as a polynomial in q, x with coefficients in $k_0[t_1, t_2, \dots, t_m]$ and equating each of these coefficients to 0, will be called the *0-system* corresponding to the polynomial φ .

4.2.1 Decidability of $\mathcal{P}_{k_0, \Lambda}$

We can check whether or not there exist in Λ values of parameters that annihilate all the coefficients of the original equation (with an operator L of the form (22)). To do this we construct the system S' of all equations of 0-systems corresponding to coefficients r_i , $i = 0, 1, \dots, \rho$, of the operator L , and apply $A_{k_0, \Lambda}$ to $(S', \emptyset)$. If the result of this applying is “yes” then the original q -difference equation with such values of parameters turns into $0 = 0$. Any polynomial is a solution of this equation.

If such values of parameters do not exist, then by Proposition 2(i) the value $l = w_q$ can be used as an upper bound on the degree of any polynomial solution. Of course, for different values of parameters we will get after their substitution into (22) different operators with different values w_q . But none of these w_q 's exceeds the value that is found for (22).

The method of undetermined coefficients can be used. Let $y_0, y_1, \dots, y_l$ be the undetermined coefficients. We get a system S of linear homogeneous algebraic equations for $y_0, y_1, \dots, y_l$ with coefficients from $k_0[q, t_1, t_2, \dots, t_m]$, and it is sufficient to recognize whether or not exist in Λ such values of $t_1, t_2, \dots, t_m$ that the system which is obtained as a result of substituting these values into S , has a non-zero solution with components in $\Lambda(q)$.

We obtain the following algorithm.

Construct S' of all equations of 0-systems corresponding to the coefficients r_i , $i = 0, 1, \dots, \rho$, of operator L , and apply $A_{k_0, \Lambda}$ to $(S', \emptyset)$; if the result is “yes”, then stop the algorithm with the answer “yes” (we will assume in the rest of the description of this algorithm that such values do not exist). Set $l = w_q$. Construct the system of linear algebraic equations for coefficients $y_0, y_1, \dots, y_l$ of an arbitrary polynomial solution of $L(y) = 0$. Let T be the matrix of this linear system (the elements of T belong to $k_0[q, t_1, t_2, \dots, t_m]$). Construct the system of algebraic equations, gathering together equations of the 0-systems of all the minors of order $l + 1$ of T ,

and apply $A_{k_0, \Lambda}$ to

$$(S, \emptyset), \quad (23)$$

where S is the constructed system.

So the problem $\mathcal{P}_{k_0, \Lambda}$ is decidable.

REMARK 3. In contrast to the q -difference case, in the differential and difference cases no independent of the values of parameters upper bound for the degree of polynomial solutions exists in general. For example the differential equation $xy' - ty = 0$ with one parameter t has the polynomial solution x^t of degree t when $t \in \mathbb{N}$. Similarly the difference equation $xy(x+1) - (x+t)y(x) = 0$ has the polynomial solution $x(x+1)\dots(x+t-1)$ of degree t when $t \in \mathbb{N}$.

4.2.2 Additional constraints

If originally an algebraic system S_1 for $t_1, t_2, \dots, t_m$ is given, then the existence of parameter values which satisfy S_1 and for which the equation $L(y) = 0$ has a non-zero polynomial solution, can be recognized by the above algorithm, provided that we use $S_1 \cup S$ instead of S in (23).

If we investigate the existence of the values of parameters which do not satisfy a non-empty system S_2 and for which the equation $L(y) = 0$ has a non-zero polynomial solution, then we use S_2 instead of $\emptyset$ in (23).

It is also possible to consider two additional systems, the first of which has to be satisfied, while the second one must not be satisfied (if it is not empty).

4.2.3 Decidability of $\mathcal{R}_{k_0, \Lambda}$

Now consider the problem $\mathcal{R}_{k_0, \Lambda}$. For a given equation $L(y) = 0$ with parameters we can use formula (21) to find $V \in \Lambda(q, x, t_1, t_2, \dots, t_m)$ (since our q -difference equation is homogeneous, we can take $w = w_q$). Then we substitute $y = zV$ into $L(y) = 0$, clear denominators and decide whether or not a non-zero polynomial solution of the resulting equation exists.

Note that the corresponding values of parameters should not annihilate the polynomial r_ρ , which is included in the denominator of (21) (but it is easy to show that there is no trouble with the case when r_0 is annihilated). We apply the algorithm from Section 4.2.2, using the system S_2 , which is the 0-system corresponding to r_ρ . If such values of parameters do not exist then set $\tilde{L} = L - r_\rho Q^\rho$, the adequate values have to satisfy the 0-system corresponding to the polynomial r_ρ , and so on.

Now we can give a description of the full algorithm. The algorithm is applicable to an equation $L(y) = 0$ and a system S_1 of algebraic equations, which has to be satisfied by the adequate values of parameters. Even if initially S_1 contains no equations ($S_1 = \emptyset$), and is satisfied by any values of parameters, then non-empty systems S_1 may appear due to recursive calls in this algorithm.

If $L = 0$ then apply $A_{k_0, \Lambda}$ to $(S_1, \emptyset)$ and stop algorithm with the obtained answer (in the rest of the description of this algorithm we will assume that $L \neq 0$). Construct the 0-system S_2 corresponding to the polynomial r_ρ . Find V by formula (21), substitute $y = zV$ into $L(y) = 0$, clearing denominators; this gives an equation $L'(z) = 0$. By the algorithm from Sections 4.2.1, 4.2.2 recognize the existence of parameter values which satisfy S_1 but not S_2 (if the latter system is not empty) and such that the equation $L'(z) = 0$

has a non-zero polynomial solution. Stop the algorithm with the answer "yes" if such values exist, otherwise apply the algorithm recursively to $\tilde{L}(y) = 0$, $\tilde{S}_1$, where $\tilde{L} = L - r_\rho Q^\rho$ and $\tilde{S}_1 = S_1 \cup S_2$.

So the problem $\mathcal{R}_{k_0, \Lambda}$ is decidable.

4.2.4 The case of a single parameter

Let there be only one parameter, denoted by t . In this case any non-empty algebraic system is equivalent to a single equation $s(t) = 0$, which can be constructed by the Euclidean algorithm. If $s(t)$ is a non-zero polynomial, then we can assume that it is square-free (otherwise we take the quotient of $s(t)$ and $\gcd(s(t), s'(t))$, where $s'(t)$ is the derivative of the polynomial $s(t)$). If both systems in the original pair are non-empty, then we obtain the pair

$$(s_1(t) = 0, s_2(t) = 0), \quad (24)$$

where each of polynomials $s_1(t)$, $s_2(t)$ is either zero or square-free. In this case

- if $s_2(t)$ is the zero polynomial, then (24) has no solution in Λ ,
- if $s_2(t) \in k_0[x] \setminus \{0\}$, but $s_1(t)$ is the zero polynomial, then the set of all solutions of (24) belonging to Λ is the set $\{\lambda \in \Lambda; s_2(\lambda) \neq 0\}$,
- if $s_1(t), s_2(t) \in k_0[x] \setminus \{0\}$, then the set of all solutions of (24) belonging to Λ is the set $\{\lambda \in \Lambda; s(\lambda) = 0\}$ where $s(t) = s_1(t)/\gcd(s_1(t), s_2(t))$.

Therefore the set of adequate values of the parameter has the form U or $\Lambda \setminus U$, where U is the set of those roots of a concrete polynomial over k_0 which belong to Λ . It is easy to see that if M_1, M_2 are sets of this form then the sets $M_1 \cup M_2$, $M_1 \cap M_2$, and $\Lambda \setminus M_1$ are of the same form.

This implies that, e.g., in the case when $m = 1$ and $k_0 = \Lambda = \mathbb{Q}$ we are able to obtain all the desired solutions of the original pair (notice that we did not include $\mathbb{Q}$ in the list (17); we will discuss more about this in Section 4.2.6).

The algorithms in Sections 4.2.1, 4.2.2, 4.2.3 are designed in such a way that if at some point it is detected that adequate parameter values exist, then the algorithms stop. For $m = 1$ these algorithms can be easily modified so that the set of all the adequate values can be presented in simple form.

4.2.5 The main statement for the case of parameters, independent of q

The reasoning given above proves the following theorem.

THEOREM 1. Let the assumptions 1 – 3, formulated in Section 4.1, be valid. Then

- (i) the question whether or not there exist adequate parameter values can be answered algorithmically, and therefore the problems $\mathcal{P}_{k_0, \Lambda}$, $\mathcal{R}_{k_0, \Lambda}$ are decidable;
- (ii) in the case of a single parameter the set of adequate parameter values has the form U or $\Lambda \setminus U$, where U is the set of those roots of a polynomial $h(t) \in k_0[t]$ which belong to Λ . The polynomial $h(t)$ can be constructed algorithmically (it can be the zero polynomial, in this case $\Lambda \setminus U = \emptyset$). This polynomial is independent of Λ .

Recall that algorithms solving the problem $\mathcal{S}_{k_0, \Lambda}$ for the fields Λ from the list (17) are known for $k_0 = \mathbb{Q}$.

4.2.6 The case $k_0 = \Lambda = \mathbb{Q}$

Let $k_0 = \Lambda = \mathbb{Q}$. It is not clear whether the problem of existence of $\tau_1, \tau_2, \dots, \tau_m \in \Lambda$ such that after substituting the values $\tau_1, \tau_2, \dots, \tau_m$ for $t_1, t_2, \dots, t_m$ in $L(y) = 0$ the resulting equation has a non-zero polynomial (rational-function) solution, is decidable. Let us show that if it is decidable then the problem of the existence of a solution with components belonging to $\mathbb{Q}$ of a given algebraic equation with integer coefficients is decidable too (the question of the decidability of the later problem is still open, the common opinion of experts is that this problem is undecidable, see, e.g., [22]). Indeed, let $P(t_1, t_2, \dots, t_m)$ be an arbitrary polynomial with integral coefficients. Then for any values $\tau_1, \tau_2, \dots, \tau_m \in \mathbb{Q}$, the indicial equation $I_0(\lambda) = 0$ (see Remark 2) of the q -difference equation

$$y(qx) - (1 + P(\tau_1, \tau_2, \dots, \tau_m))y(x) = 0 \quad (25)$$

is $\lambda - 1 - P(\tau_1, \tau_2, \dots, \tau_m) = 0$. This indicial equation has a root of the form q^h , $h \in \mathbb{Z}$, only if $P(\tau_1, \tau_2, \dots, \tau_m) = 0$. Then $h = 0$, and the q -difference equation (25) is satisfied, e.g., by the polynomial $y(x) = 1$.

4.2.7 On possible values of q

If q is an additional variable besides x , then q is transcendental over any of the fields (17). When $k_0 = \mathbb{Q}$ the previous results are valid also if q is a transcendental number (i.e., $q \in \mathbb{C} \setminus \overline{\mathbb{Q}}$ or, in the real case, $q \in \mathbb{R} \setminus \overline{\mathbb{Q}}$), and Λ is one of $\overline{\mathbb{Q}}$, $\mathbb{R} \cap \overline{\mathbb{Q}}$.

4.3 Inhomogeneous equations

4.3.1 Polynomial right-hand sides

In the Introduction we listed some concrete undecidable problems, connected with differential and difference linear homogeneous equations with numerical parameters. We described above algorithms for solving those problems in the case of q -difference equations. Similar algorithms can be applied in the case of linear *inhomogeneous* q -difference equations, when the right hand side f is a polynomial in x with coefficients in $k_0[q, t_1, t_2, \dots, t_m]$. It follows from (7) that we can use $\max\{\deg_x f, w_q\}$ as an upper bound for degrees of polynomial solutions. For constructing rational-function solutions we can use the algorithm from Section 4.2.3 using the same bounding rule for polynomial solutions.

Checking the existence of polynomial solutions, we obtain an inhomogeneous system of linear algebraic equations whose matrix T and right-hand sides consist of elements of $k_0[q, t_1, t_2, \dots, t_m]$. By means of algorithms considered above we can recognize whether or not there exist parameter values annihilating the right-hand side of this system such that the corresponding homogeneous system has a non-zero solution. The condition (on parameters values) that the right-hand side of the system is not annihilated we call the *inhomogeneity condition*. Suppose that the inhomogeneity condition is satisfied. Using, e.g., step-by-step consideration of minors and Kronecker-Capelli's theorem, we can recognize whether there exist parameter values for which the system is compatible (there exists a non-zero minor of some order n of the matrix T while any minor of order $n + 1$ of the augmented matrix $\tilde{T}$ are equal to zero). This analysis can be done by the algorithm $A_{k_0, \Lambda}$. In the case of a single parameter the set of adequate parameter values can be presented as in Section 4.2.4.

4.3.2 Parametric summation

If k_0, Λ are such that the problem $\mathcal{R}_{k_0, \Lambda}$ is decidable in the inhomogeneous case then, e.g., the parametric problem of q -hypergeometric summation is decidable also, and in the q -difference case it is possible to consider a parametric version of Gosper's algorithm, since in this algorithm one can use the universal factors instead of the special Gosper form of rational functions representation. (Parametric versions of algorithms that are based on Gosper's algorithm [21] probably exist, too; see, e.g., [9, Sect. 3].) It is also possible to propose q -difference version of the accurate integration (summation) algorithm [7, 8]. In the one-parametric case we not only can *recognize the existence* of adequate values of parameters, but can also *find* them. However one should not forget that the algorithms discussed above have high complexity. As mentioned, the aim of this paper is only to establish decidability of some algorithmic problems "in principle".

5. WHEN PARAMETERS DEPEND ON Q

Let the assumptions 1 and 3, formulated in Section 4.1, be valid. We will consider algorithmic problems similar to $\mathcal{P}_{k_0, \Lambda}$ and $\mathcal{R}_{k_0, \Lambda}$ (the homogeneous case) investigated above, allowing parameter values belong to $\Lambda[q]$ or $\Lambda(q)$. From this point on we will consider the problems

$$\mathcal{P}_{k_0, \Lambda[q]}, \quad \mathcal{R}_{k_0, \Lambda[q]} \quad (26)$$

and

$$\mathcal{P}_{k_0, \Lambda(q)}, \quad \mathcal{R}_{k_0, \Lambda(q)}. \quad (27)$$

In (26) parameter values belong to the ring $\Lambda[q]$, in (27) they belong to the field $\Lambda(q)$.

5.1 Two theorems of J. Denef

In our investigation of problems (26), (27) the key role will be played by two theorems of Denef [15]. Before formulating them we introduce two notions following [15]: Let R be a commutative ring with unity and let R' be a subring of R . We say that the *diophantine problem for R with coefficients in R'* is undecidable (decidable) if there exists no (an) algorithm to decide whether or not a polynomial equation (in several variables) with coefficients in R' has a solution in R .

The following results are proved in [15]:

THEOREM A. *Let R be an integral domain of characteristic zero; then the diophantine problem for $R[T]$ with coefficients in $\mathbb{Z}[T]$ is undecidable. ($R[T]$ denotes the ring of polynomials over R , in one variable T .)*

THEOREM B. *Let K be a formally real field, i.e., -1 is not the sum of squares in K . Then the diophantine problem for $K(T)$ with coefficients in $\mathbb{Z}[T]$ is undecidable. ($K(T)$ denotes the field of rational functions over K , in one variable T .)*

As a consequence of Theorems A, B we obtain the following:

The diophantine problem for $\Lambda[q]$ with coefficients in $\mathbb{Z}[q]$ is undecidable. If the field Λ is formally real, then the diophantine problem for $\Lambda(q)$ with coefficients in $\mathbb{Z}[q]$ is also undecidable.

5.2 Undecidability in the case of parameters depending on q

Now we engage in problems (26), (27) in earnest.

LEMMA 1. Let $P(t_1, t_2, \dots, t_m)$ be an arbitrary polynomial with coefficients in $\Lambda[q]$ (in particular, in $\mathbb{Z}[q]$). Then the equation

$$y(qx) - (1 + P^2(t_1, t_2, \dots, t_m))y(x) = 0 \quad (28)$$

with some rational functions (in particular, polynomials) $t_1 = \tau_1(q)$, $t_2 = \tau_2(q)$, $\dots$, $t_m = \tau_m(q)$ over Λ has a non-zero solution y in $\Lambda(q)(x)$ iff $P(\tau_1(q), \tau_2(q), \dots, \tau_m(q)) = 0$.

Proof. Since q is transcendental over $\Lambda(x)$, q can be considered as a variable. If $P(\tau_1(q), \tau_2(q), \dots, \tau_m(q)) \in \Lambda(q) \setminus \{0\}$ has the form of a fraction $\frac{f(q)}{g(q)}$ with relatively prime polynomials $f(q), g(q)$ over Λ , then

$$I_0(\lambda) = \lambda - 1 - \frac{f^2(q)}{g^2(q)}$$

in the corresponding indicial equation. But the equation $I_0(\lambda) = 0$ has no roots of the form q^h , $h \in \mathbb{Z}$ (see Remark 2). Indeed, $h \neq 0$, because otherwise $\frac{f(q)}{g(q)}$ is the zero rational function. If $h > 0$, then we would have the equality $(q^h - 1)g^2(q) = f^2(q)$ in $\Lambda[q]$. However the irreducible factor $q - 1$ appears in the left-hand side with an odd exponent, while in the right-hand side it appears with an even exponent – a contradiction. If $h < 0$ then for $h_0 = -h$ we have $-(q^{h_0} - 1)g^2(q) = f^2(q)q^{h_0}$. This is impossible for the same reasons.

If $P(\tau_1(q), \tau_2(q), \dots, \tau_m(q)) = 0$, then the equation (28) has, e.g., the solution that is identically equal to 1. $\square$

THEOREM 2. The problems (26) are undecidable. In addition, if Λ is a formally real field then the problems (27) are undecidable as well.

Proof. By the consequence of Theorems A, B formulated in Section 5.1, and by Lemma 1. $\square$

Let $k_0 = \mathbb{Q}$. If q is a variable, $\Lambda \in \{\mathbb{C}, \mathbb{R}, \mathbb{Q}, \overline{\mathbb{Q}}, \mathbb{R} \cap \overline{\mathbb{Q}}\}$ then the problems (26) are undecidable. The same is true if q is a transcendental number and $\Lambda \in \{\mathbb{Q}, \overline{\mathbb{Q}}, \mathbb{R} \cap \overline{\mathbb{Q}}\}$. In turn, the problems (27) are undecidable if, e.g., q is a variable and $\Lambda \in \{\mathbb{R}, \mathbb{Q}, \mathbb{R} \cap \overline{\mathbb{Q}}\}$, or if q is a transcendental number and $\Lambda \in \{\mathbb{Q}, \mathbb{R} \cap \overline{\mathbb{Q}}\}$.

5.3 The case $\Lambda = \mathbb{C}$

It is not clear whether or not the problems (27) are decidable when, e.g., $k_0 = \mathbb{Q}$, $\Lambda = \mathbb{C}$ (q is a variable). However it follows from Lemma 1 that if at least one of them is decidable then the diophantine problem for $\mathbb{C}(q)$ with coefficients from $\mathbb{Z}[q]$ is decidable as well. Notice that the latter problem is still open, but the common opinion of experts is such that it is undecidable — we again refer to the survey [22].

6. REFERENCES

- [1] S. A. Abramov. Rational solutions of linear difference and q -difference equations with polynomial coefficients. *Programming and Comput. Software*, **21**, No 6, 273–278 (1995).
- [2] S. A. Abramov. On an undecidable problem connected with differential and difference equations. *Proceedings of XIII International conference on differential equations (Erugin readings - 2009)*, May 26–29, 2009, Pinsk, Belarus, p. 118 (in Russian) (2009).
- [3] S. A. Abramov. On an undecidable problem related to difference equations with parameters, *Programming and Computer Software*, **36**, No. 2 (2010).
- [4] S. A. Abramov. A direct algorithm to compute rational solutions of first order linear q -difference systems, *Discrete Math.*, **246**, 3–12 (2002).
- [5] S. A. Abramov, M. Bronstein. Hypergeometric dispersion and the orbit problem, *ISSAC'00 Proceedings*, 8–13 (2000).
- [6] S. A. Abramov, M. Bronstein and M. Petkovšek. On polynomial solutions of linear operator equations, *ISSAC'95 Proceedings*, 290–296 (1995).
- [7] S. A. Abramov, M. van Hoeij. A method for the integration of solutions of Ore equations, *ISSAC'97 Proceedings*, 172–175 (1997).
- [8] S. A. Abramov, M. van Hoeij. Integration of solutions of linear functional equations, *Integral Transformations and Special Functions*, **8**, No 1-2, 3–12 (1999).
- [9] S. A. Abramov, S. P. Polyakov. Improved universal denominators, *Programming and Computer Software*, **33**, No. 3, 123–137 (2007).
- [10] G. E. Andrews. *The Theory of Partitions*. Encyclopedia of Mathematics and its Applications, Vol. 2, Addison-Wesley, Reading, Mass., 1976.
- [11] G. E. Andrews. *q-Series: Their Development and Application in Analysis, Number Theory, Combinatorics, Physics, and Computer Algebra*. CBMS Regional Conference Series, No. 66, AMS, R.I., 1986.
- [12] M. Barkatou. Rational solutions of matrix difference equations: problem of equivalence and factorization, *ISSAC'99 Proceedings*, 277–282 (1999).
- [13] D. Boucher. About the polynomial solutions of homogeneous linear differential equations depending on parameters, *ISSAC'99 Proceedings*, 261–268 (1999).
- [14] B. Buchberger. Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory. In: *Recent Trends in Multidimensional System Theory*, D. Reidel, Dordrecht, 1985.
- [15] J. Denef. The diophantine problem for polynomial rings and fields of rational functions, *Transactions of the American Mathematical Society*, **242**, 391–399 (1978).
- [16] L. van den Dries. Alfred Tarski's elimination theory for real closed fields, *J. Symbolic Logic*, **53**, 7–19 (1988).
- [17] J. von zur Gathen, J. Gerhard. *Modern Computer Algebra* (Second Edition). Cambridge University Press, 2003.
- [18] M. van Hoeij. Rational solutions of linear difference equations, *ISSAC'98 Proceedings*, 120–123 (1998).
- [19] Yu. V. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, Cambridge, MA, 1993.
- [20] B. Mishra. *Algorithmic Algebra*. Springer-Verlag, 1993.
- [21] M. Petkovšek, H. S. Wilf, D. Zeilberger. $A = B$, A K Peters, 1996.
- [22] T. Pheidas and K. Zahidi. Undecidability of existential theories of rings and fields: A survey. *Contemporary Mathematics*, **270**, 49–106 (2000).
- [23] Maple online help: <http://www.maplesoft.com/support/help/>

Iterative Toom-Cook Methods For Very Unbalanced Long Integer Multiplication

Alberto Zanoni

Centro Interdipartimentale “Vito Volterra”
Università degli Studi di Roma “Tor Vergata”
Via Columbia 2, 00133 Roma (Italy)
zanoni@volterra.uniroma2.it

ABSTRACT

We consider the multiplication of long integers when one factor is much larger than the other one. We describe an iterative approach using Toom-Cook unbalanced methods, which results in the evaluation of the smaller integer only once. The particular case of Toom-2.5 is considered in full detail. A further optimization depending on the parity of the shortest operand evaluation in 1 is also described. A comparison with GMP library is also presented.

Categories and Subject Descriptors

F.2.1 [Analysis of algorithms and program complexity]: Numerical algorithms and problems—*Computations on polynomials*; G.1.1 [Numerical analysis]: Interpolation—*Interpolation formulas*; G.2.3 [Discrete mathematics]: Applications; I.1.2 [Computing methodologies]: Algorithms—*Algebraic algorithms*

General Terms

Algorithms, Performance, Theory

Keywords

Long integer multiplication, Toom-Cook, interpolation

1. INTRODUCTION

Starting with the works of Karatsuba [9], Toom [13] and Cook [5], who found methods for the lowering of the asymptotic complexity for polynomial multiplication from $O(n^2)$ to $O(n^e)$, where $1 < e \leq \log_2 3$, many efforts have been made to find optimized implementations in arithmetic software packages [10], [7], [8].

The family of Toom-Cook (Toom, for short) methods is an infinite set of algorithms (Toom-3, Toom-4, etc. – Karatsuba may be identified with Toom-2). The original family was generalized by Bodrato and Zanoni in [4] by considering

unbalanced operands as well – that is, polynomials with different degrees – with the so-called Toom- $(k + 1/2)$ methods (Toom-2.5, Toom-3.5, etc.) and with the unbalanced use of classic methods.

Each of them may be viewed as solving a polynomial interpolation problem, with base points not specified *a priori*, from which the matrix to be inverted is derived.

In a software implementation, a set of basic operations (typically sums, subtractions, bit shiftings, multiplications and divisions by small numbers, etc.) is given. Practically, it's a set of very efficiently implemented basic functions in a particular computer language. The idea is to use them both to evaluate factors in the base points and invert the resulting matrix step by step by using elementary row operations.

When Toom methods are applied to long integer multiplication, carries and borrows enter the process, and the implementation code must include these features. Release 5.0.0 of GMP library [8], our reference, implements many Toom methods, for balanced and limitedly unbalanced factors (for latest results, see [2]). For *very* unbalanced factors, either an iterative approach or, if factors are very long, Schönhage and Strassen's FFT-based method (see [12]) is instead used.

In the unbalanced case, the idea of splitting the long factor in smaller pieces to be handled separately with the short factor subsequently combining the results was proposed by Schönhage ([10], 57, exercise 4.3.3-13). The feature of saving/reusing computations (evaluations, in our case) is present e.g. in [11], [6].

In this work the general use of unbalanced Toom methods for iterative multiplication applied to very unbalanced factors is described, with an *ad hoc* optimization. In particular, we explain in full detail the basic and iterative Toom-2.5 method (the smallest unbalanced case). We also will find by considering parity a further optimization.

2. TOOM-COOK ALGORITHM

Below is a brief summary of the Toom- k multiplication algorithm for natural numbers. Let $u, v \in \mathbb{N}$: to compute the product $u \cdot v = w \in \mathbb{N}$, follow the five below indicated steps.

Splitting : Fix an appropriate basis $B \in \mathbb{N}$ and represent the two operands by two homogeneous polynomials $a, b \in \mathbb{N}[x, h]$ with degree d_1, d_2 respectively and coefficients $0 \leq a_i, b_i < B$ (base B expansion). In computer

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

science, usually B is a power of two.

$$a(x, h) = \sum_{i=0}^{d_1} a_i x^i h^{d_1-i} \quad ; \quad b(x, h) = \sum_{i=0}^{d_2} b_i x^i h^{d_2-i}$$

One has $u = a(B, 1)$ and $v = b(B, 1)$. Let $c(x, h) = a(x, h)b(x, h)$, with $\deg(c) = d_1 + d_2$. For the classical Toom- k method one has $d_1 = d_2 = k - 1$, while in general, for possibly unbalanced operands, $d_1 + d_2 = 2(k - 1)$, k being an arbitrary multiple of $1/2$.

Evaluation : Choose $2k - 1$ values $v_i = (v'_i, v''_i) \in \mathbb{Z}^2$ (the corresponding homogeneous value is v'_i/v''_i , with $(1, 0)$ usually represented by ∞) with v'_i and v''_i coprime and $v_i \neq \pm v_j$ for $i \neq j$: evaluate both operands on all of them, obtaining $a(v_i)$, $b(v_i)$.

Recursion : Compute $w_i = a(v_i) \cdot b(v_i)$ recursively. Let $\mathbf{w} = (w_i)$ be the resulting vector.

Interpolation : Solve the interpolation problem $c(v_i) = w_i$ inverting the pseudo-Vandermonde matrix A_k generated by the v_i values, computing $\mathbf{c} = A_k^{-1}\mathbf{w}$, where $\mathbf{c} = (c_i)$ is the vector of $c(x, h)$ coefficients.

Recomposition : Once all coefficients are computed, it's sufficient to evaluate back $w = c(B, 1)$.

Standard analysis shows that Toom- k method complexity is $O(n^{\log_k(2k-1)})$. The multiplicative constant hidden by the $O(\cdot)$ notation absorbs the complexity of the first two and last two phases. In order to minimize it, an accurate choice of v_i values and of the operations sequence for Evaluation, Interpolation and Recomposition phases helps in reducing the extra overhead. Studies and analyses concerning this can be found in [4], [3], [16].

Example : The matrices $A_{2.5}$ and A_3 , obtained by the interpolation values $\{(1, 0), (-1, 1), (1, 1), (0, 1)\}$ and $\{(1, 0), (2, 1), (-1, 1), (1, 1), (0, 1)\}$, respectively, are

$$A_{2.5} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad ; \quad A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 16 & 8 & 4 & 2 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

3. TOOM-2.5 METHOD DESCRIPTION

While in the following section we'll describe the iterative use of unbalanced Toom methods, we report here all details of the basic version of the smallest one, Toom-2.5. For the sake of simplicity, in this and in the following section non-homogeneous notation will be used. In this case we set $\nu' = \lceil \log_2 \nu \rceil + 1$, $\nu = \lceil \nu'/2 \rceil$ and $B = 2^\nu$, obtaining

$$a(x) = a_2 x^2 + a_1 x + a_0 \quad ; \quad b(x) = b_1 x + b_0 \\ c(x) = c_3 x^3 + c_2 x^2 + c_1 x + c_0$$

We recall that, because of the recursive use, the coefficients a_0, a_1, a_2, b_0 and b_1 are long integers as well, with bit length $\leq \nu$ (a_2 and b_1 bit lengths may be strictly smaller than ν). There are only two non trivial evaluations to compute, corresponding in practice to set $x = \pm 1$.

3.1 Classical version

Evaluation phase is completed with 5 additions/subtractions, as follows:

$$\begin{aligned} a^- &= a_2 + a_0 & b^+ &= b_0 + b_1 \\ a^+ &= a^- + a_1 & b^- &= b_0 - b_1 \\ a^- &= a^- - a_1 \end{aligned} \quad ;$$

Computing the 4 needed values by multiplying corresponding evaluations we may set up the interpolation problem:

$$w_i = a(v_i)b(v_i) = c(v_i) \quad ; \quad i = 0, \dots, 3 \implies$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} w_3 \\ w_2 \\ w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} a_2 b_1 \\ a^- b^- \\ a^+ b^+ \\ a_0 b_0 \end{pmatrix}$$

As the bit length of w_i is 2ν (it may be smaller for w_3), we indicate with prefixes H and L their high (most significant) and low (least significant) parts: $w_i = H_i B + L_i$. Note moreover that $c_3 = w_3$ and $c_0 = w_0$. Beginning the interpolation, with two algebraic sums and a division by 2 (right bit shift, indicated with $\gg$) one destructively has what we call the “decoupling” step:

$$\begin{aligned} w_2 &= w_2 + w_1 &\implies &\text{Addition} \\ w_2 &= w_2 \gg 1 &\implies &\text{Right shift (division by 2)} \\ w_1 &= w_1 - w_2 &\implies &\text{Subtraction} \end{aligned}$$

If at this point Interpolation and Recomposition phases are “mixed”, it is possible to save a $O(\nu)$ -addition. Consider infact the final recomposition at this point: formally, with new values of w_1 and w_2 after decoupling, we have

$$c(x) = (w_3)x^3 + (w_2 - w_0)x^2 + (w_1 - w_3)x + w_0$$

but there are overlaps: we may infact rewrite $c(x)$ as follows.

$$\begin{aligned} c(x) &= (H_3)x^4 + (H_2 + (L_3 - H_0))x^3 + \\ &\quad (L_2 - H_3 + H_1 - L_0)x^2 + (L_1 - (L_3 - H_0))x + L_0 \end{aligned}$$

Note that $L_3 - H_0$ occurs two times, and can be computed just once. Supposing that an operation (with linear complexity) in the Interpolation phase has double cost with respect to operations in Evaluation phase – as operands have double length – 15 additions and 2 shifts of complexity $O(\nu)$ each are needed. It is noted that the technical details of carries and borrows need to be considered, however they are not given here.

3.2 Even version

Sometimes it is also possible to use another version of Toom-2.5, by applying the so-called “divide by 2” technique introduced in [15]. The technique is based on the simple observation that

$$\frac{c(v_i)}{C} = \frac{a(v_i)b(v_i)}{C} = \frac{a(v_i)}{C}b(v_i) = a(v_i)\frac{b(v_i)}{C}$$

for whatever constant $C \in \mathbb{N}$ dividing exactly both members of whatever pair of above equalities. Now, take $C = 2$ for Toom-2.5 and test if $b_0 \equiv b_1 \pmod{2}$ – just compare their least significant bits. If it is the case, then $b^+ = b_0 + b_1$ (and $b^- = b_0 - b_1$ as well) is even, and we can compute the “divided by 2” products

$$w'_1 = \frac{w_1}{2} = a^+ \frac{b^+}{2} \quad ; \quad w'_2 = \frac{w_2}{2} = a^- \frac{b^-}{2}$$

realizing the decoupling step with just two algebraic additions. This can also be done if $a_0 + a_2 \equiv a_1 \pmod{2}$ (a^+ and a^- as well are even). In the below pseudo-code, t is a

temporary variable.

$$\left. \begin{array}{l} \text{Even evaluations : } a(x) \\ a^+ = a_2 + a_0 \quad b^+ = b_0 + b_1 \\ a^+ = a^+ + a_1 \quad b^- = b_0 - b_1 \\ a^+ = a^+ \gg 1 \quad a^- = a^+ - a_1 \\ \\ \text{Even evaluations : } b(x) \\ a^- = a_2 + a_0 \quad b^- = b_0 - b_1 \\ a^+ = a^- + a_1 \quad b^- = b^- \gg 1 \\ a^- = a^- - a_1 \quad b^+ = b^- + b_1 \end{array} \right\} \text{Decoupling}$$

$$\begin{array}{l} t = a^- b^- \\ w'_1 = a^+ b^+ \\ w'_2 = w'_1 + t \\ w'_1 = w'_1 - t \end{array}$$

Note that the shift operation was moved in the evaluation phase, and its cost is halved. With a probability of 75 %, the number of $O(\nu)$ shifts is then 1, not 2, so that the average linear complexity of Toom-2.5 is (A = addition, S = shift)

$$\mathcal{L}_{2.5} = 15A + \frac{1}{4}(2S) + \frac{3}{4}S = 15A + \frac{5}{4}S$$

Thoretically speaking [1], it is possible to consider the “full” even version (100% of cases) by splitting in a different way the shortest factor when $b_0 \equiv b_1 + 1 \not\equiv b_1 \pmod{2}$. Let

$$b_1 2^\nu + b_0 = b_1 2^\nu + 2^\nu - 2^\nu + b_0 = (b_1 + 1)2^\nu - (2^\nu - b_0)$$

Setting $b'_1 = b_1 + 1$, $b'_0 = 2^\nu - b_0$ we can consider an alternative representation of the second factor: $b'(x) = b'_1 x - b'_0$. As $b'_1 \equiv b_1 + 1 \pmod{2}$ and $b'_0 \equiv b_0$, now $b'_0 \equiv b'_1 \pmod{2}$. The “divide by 2” technique could then be applied, but b'_0 management asks for one subtraction more, so that the practical benefit is lost.

4. ITERATIVE TOOM-COOK METHODS

In [14] an approach to a multiple use of Karatsuba (i.e. Toom-2) method was proposed. Inspired by those ideas, we present here an iterative approach for generic very unbalanced Toom methods. We first describe the particular case of iterative Toom-2.5, and then the general one. Let $d_1 = d$ be much larger than $d_2 = 1$: we have

$$a(x) = \sum_{i=0}^d a_i x^i \quad ; \quad b(x) = b_1 x + b_0$$

so that

$$\begin{aligned} c(x) = \sum_{i=0}^{d+1} c_i x^i &= (a_d b_1) x^{d+1} + (a_d b_0 + a_{d-1} b_1) x^d + \dots \\ &\dots + (a_1 b_0 + a_0 b_1) x + (a_0 b_0) \end{aligned}$$

with, for $i = 1, \dots, d$,

$$c_{d+1} = a_d b_1 \quad ; \quad c_0 = a_0 b_0 \quad ; \quad c_i = a_i b_0 + a_{i-1} b_1 \quad (1)$$

4.1 Iterative Toom-2.5

Let $m = \lfloor (d+1)/3 \rfloor$ and split $a(x)$ into sub-polynomials (“sections”) of degree 2; i.e. set $y = x^3$ and consider $a(x)$ as a bivariate polynomial $\bar{a}(x, y) = \sum_j a_j(x) y^j = \sum_j (a_{3j+2} x^2 + a_{3j+1} x + a_{3j}) y^j$:

$$\begin{aligned} a(x) &= a'(x) x^{3m} + (a_{3m-1} x^2 + a_{3m-2} x + a_{3m-3}) x^{3(m-1)} + \\ &\dots + (a_5 x^2 + a_4 x + a_3) x^3 + (a_2 x^2 + a_1 x + a_0) \end{aligned}$$

Note that we may have a “border” effect when the most significant part of $a(x)$, indicated here with $a'(x)$, has degree

smaller than 2 (it may be $a'(x) = 0$ as well). Applying Toom-2.5 method to $a_j(x)$ and $b(x)$, we may reconstruct sectionwise the final result.

Note that for each product $a_j(x)b(x)$ the second factor $b(x)$ is fixed, and therefore we can compute the values b^+ and b^- just once, using them for every product. At step j , Toom-2.5 gives the four values

$$a_{3j+2} b_1 \quad , \quad c_{3j+2} \quad , \quad c_{3j+1} \quad , \quad a_{3j} b_0$$

The coefficients c_{3j} , with $j > 0$, can be obtained simply by adding the products $a_{3(j-1)+2} b_1$, obtained at step $j-1$, and $a_{3j} b_0$, obtained at step j (when $j = 0$, we directly have $c_0 = a_0 b_0$, with no extra addition, and similarly for c_{d+1} , if $\deg(a'(x)) = 2$). If $\deg(a'(x)) < 2$, recursively compute $c'(x) = a'(x)b(x)$, using the most appropriate multiplication method, finally obtaining the remaining highest part of $c(x)$, whose less significant part must be combined with the most significant part of the precedent section.

Considering additions (all sections require 15) the first one ($j = 0$) is like the (classical or even) Toom-2.5 case, while for the others ($j = 1, \dots, m-1$) there are two less, as it is not necessary to reevaluate b . However the recomposition requires two more. For the number of shift operations, there are three cases to consider:

1. $b^+ \equiv b^- \equiv 0 \pmod{2}$: even Toom-2.5 can be used for all m sections, and just *one* single shift is needed.
2. $b^+ \equiv b^- \equiv 1 \pmod{2}$, $a_j^+ \equiv a_j^- \equiv 0 \pmod{2}$: even Toom-2.5 can be used for section j .
3. $b^+ \equiv b^- \equiv 1 \pmod{2}$, $a_j^+ \equiv a_j^- \equiv 1 \pmod{2}$: classical Toom-2.5 must be used for section j .

Considering a uniform distribution probability for operands, the probability of case 1, for which one single shift is needed, is 50 % = 1/2. In the complementary event, (cases 2 and 3), the conditional probability is uniform as well, so that the average number of shifts for a single section is $1 \cdot 1/2 + 2 \cdot 1/2 = 3/2$, as one $O(\nu)$ shift is needed in case 2 and one $O(2\nu)$ shift in case 3. As there are m sections, the average number of shifts $S(m)$ is

$$S(m) = 1 \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{3}{2} m = \frac{3m+2}{4}$$

The average linear complexity $\mathcal{L}_{2.5}^{(m)}$ for all sections and the total complexity $\mathcal{L}_{2.5}(d)$ are then

$$\mathcal{L}_{2.5}^{(m)} = (15m)A + \frac{3m+2}{4}S \quad ; \quad \mathcal{L}_{2.5}(d) = \mathcal{L}_{2.5}^{(\lfloor (d+1)/3 \rfloor)} + \mathcal{L}'_{2.5}(d)$$

where $\mathcal{L}'_{2.5}(d)$ is the residual complexity due to the “border” extra component $a'(x)b(x)$, depending on $a'(x)$.

Note that, contrary to Toom-2.5, in this “iterative” case the full Toom-2.5 even version could be effectively used, as its overhead acceptable considering its enhanced benefit. In this situation only one shift (and one extra subtraction) is needed, independently from m , so that $S(m) = 1$. However, there are some technical consideration that may limit its effectiveness:

- the possibly different length of b'_1 (b'_0) w.r.t. b_1 (b_0).
- the augmented temporary memory need.
- the more sign distinctions to be made, both in Evaluation and Interpolation phase, for each section. Infact, the Evaluation-Multiplication in ∞ ($a_2 b'_1$) is positive, in 0 ($-a_0 b'_0$) is negative, while in 1 and -1 can be negative or positive.

R				S			
		$a_0 + a_1 + a_2$	$a_0 + a_2$ $a_0 + a_2$ $a_0 + a_2$ $a_0 - a_1 + a_2$			H_p	L_p
	H_2	L_2		H_1	L_1	H_p	L_p
	$H_2 + H_1$	$L_2 + L_1$		H_1	L_1	H_p	L_p
	$H_2 = \frac{H_2 + H_1}{2}$	$L_2 = \frac{L_2 + L_1}{2}$		H_1	L_1	H_p	L_p
	H_2	L_2		H_1	L_1	H_p	L_p
...	H_2	H_0	L_0	$H_1 = H_1 - H_2$	$L_1 = L_1 - L_2$	H_p	L_p
	H_2	H_0	L_0	$H_1 + L_2$	L_1	H_p	L_p
	H_2	H_0	$L_0 + L_p$	$H_1 + L_2 - L_0$	L_1	H_p	L_p
	H_2	H_0	$L_0 + L_p$	$H_1 + L_2 - L_0$	L_1	H_p	L_p
	H_2	H_0	$L_0 + L_p$	$H_1 + L_2 - L_0$	$L_1 + H_p$		
	H_2	H_0	$L_0 + L_p$	$H_1 + L_2 - L_0$	$L_1 + H_p$	H_3	L_3
	H_2	$L_3 - H_0$	$L_0 + L_p$	$H_1 + L_2 - L_0$	$L_1 + H_p$	H_3	
	H_2	$L_3 - H_0$	$L_0 + L_p$	$H_1 + L_2 - L_0$	$L_1 + H_p$	H_3	$H_2 + L_3 - L_0$
		$L_1 + H_p - L_3 + H_0$	$L_0 + L_p$	$H_1 + L_2 - L_0$		H_3	$H_2 + L_3 - L_0$
	$H_1 + L_2 - L_0 - H_3$	$L_1 + H_p - L_3 + H_0$	$L_0 + L_p$			H_3	$H_2 + L_3 - L_0$

Figure 1: Schema for iterative Toom-Cook method: generic case

Because of the complications of the C code resulting from the considerations listed above, only the results given by our implementation of the plain (iterative) even version are presented.

For the number $\mathcal{M}_c(d)$ of multiplications using the “classic” method (see equation 1), there is one product for c_0 and c_{d+1} , and two for all other c_j 's, $1 \leq j \leq d$. With Toom-2.5 iterative method, there are four multiplications every three coefficients, so that $\mathcal{M}_{2.5}(d)$ is lowered – as before, we indicate with $\mathcal{M}'(d)$ the $O(1)$ number of “remaining” multiplications due to $a'(x)$.

$$\mathcal{M}_c(d) = 2(d+1) \quad ; \quad \mathcal{M}_{2.5}(d) = 4 \left\lfloor \frac{d+1}{3} \right\rfloor + \mathcal{M}'(d)$$

Their ratio and the corresponding gain are then

$$R_{2.5}(d) = \frac{\mathcal{M}_{2.5}(d)}{\mathcal{M}_c(d)} \simeq \frac{4}{3} \cdot \frac{d+1}{2(d+1)} \xrightarrow{d \rightarrow \infty} \frac{2}{3}$$

$$G_{2.5}(d) = 1 - R_{2.5}(d) \xrightarrow{d \rightarrow \infty} \frac{1}{3} \simeq 33\%$$

4.2 Iterative Toom-k

For the generic Toom- k method (k can be whatever multiple of $1/2$), we consider its most unbalanced version, giving us sections of $a(x)$ with degree equal to $2k-3$. The schema is similar to the one used in the iterative Toom-2.5 case: evaluations of $b(x)$ can be done once, and the product of section $a_j(x)$ and $b(x)$ gives now

$$a_{(2k-2)j+2k-3}b_1, c_{(2k-2)j+2k-3}, \dots, c_{(2k-2)j+1}, a_{(2k-2)j}b_0$$

where the first and last product have to be recombined with last product of the preceding section and the first product of the following section, respectively, to obtain the two extremal coefficients.

We have $m = \left\lfloor \frac{d+1}{2k-2} \right\rfloor$ (set $y = x^{2k-2}$), and the number of multiplications is now

$$\mathcal{M}_k = (2k-1) \left\lfloor \frac{d+1}{2(k-1)} \right\rfloor + \mathcal{M}'(d)$$

with a gain

$$G_k(d) \simeq 1 - \frac{2k-1}{2(k-1)} \frac{d+1}{2(d+1)} \xrightarrow{d \rightarrow \infty} 1 - \frac{2k-1}{4(k-1)} \xrightarrow{k \rightarrow \infty} 50\%$$

For small values of k , *ad hoc* balanced and unbalanced versions of Toom- k methods are considered in GMP: each one has a different specific and specialized evaluation and interpolation phase. For asymptotically big k values, Toom- k method are simply not used in practice, because of the more effective FFT method. It is therefore not possible to present a general treatment of linear complexity of their iterative counterparts here.

5. ITERATIVE TOOM-2.5 METHOD: IMPLEMENTATION

We implemented C code for iterative Toom-2.5 method, available on request, by using GMP library. GMP packs bits in blocks, called *limbs* (typical cases are limbs with 32 or 64 bits). Our implementation needs a $(4\nu+2)$ -limbs scratch space: the software was compiled with gcc 4.3.2 on an Intel Core 2 Duo (3 GHz) processor machine.

For simplicity we report in figure 1 a schema of the needed operations only for the generic section for classical Toom-2.5, for simplicity. Column width is not related to data bit length, and empty entries refer to memory space that can be used (values indicated before empty spaces in each column are no more needed: for example, $a_0 + a_1 + a_2$ in the second line of R is used to compute H_1 , L_1 in the third line of S : then it's useless and its memory space can be freed/recycled).

For even Toom-2.5 cases and all technical details concerning carries and borrows treatment the code is available for the interested reader. We indicate with S the scratch space and with R the area in which the current part of the result should “appear”. Actually 5 ν -parts should be present in R , but we split them: the least significant three parts in R and the two most significant ones in S , so that the possible final recomposition if $\deg(a') < 2$ can be straightforwardly done.

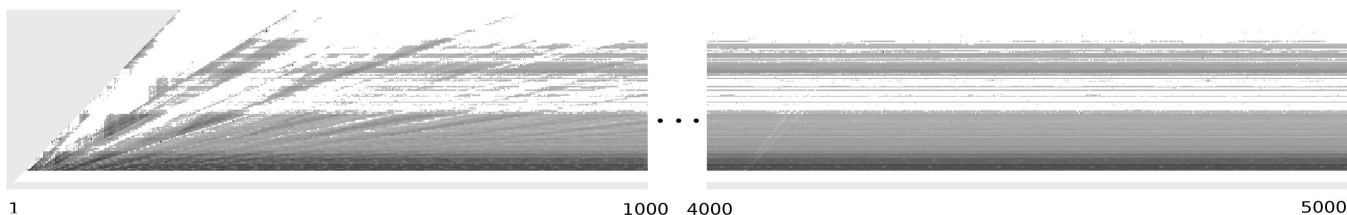


Figure 2: Iterative Toom-2.5 vs. GMP 5.0.0. Gains : Average 5.94 % ; Max 33.53 %

At the beginning, we indicate with H_p and L_p (for “precedent”) the values H_3 and $H_2 + L_3 - H_0$ (the most significant ones), respectively, of the result given by Toom-2.5 applied to the precedent section. The evaluated values b^+ and b^- are saved in the highest part of the memory area that will contain the result. At the end, (the first three parts of) R and (the first two parts of) S contain the correct values for the current section, and the whole process can be iterated.

Figure 2 shows the behavior of iterative Toom-2.5 with respect to GMP 5.0.0. On x and y axes we report the number of limbs of u (1 – 1000 ; 4000 – 5000) and v (1 – 180), respectively. Inside the region of applicability of iterative Toom-2.5 method – the infinite triangle between x -axis and the line $y = 2x/3$ – the darker the point, the bigger the gain with respect to GMP: white indicates that GMP is faster.

Iterative Toom-2.5 method is particularly effective in a well localized bottom horizontal strip, immediately above a thin white one, given by classical high-school multiplication method, applicable when one factor is small (below a certain threshold). On our architecture, in the region given in the figure, we obtained an average percentage gain (counting only cases in which iterative Toom-2.5 is faster than GMP) of 5.94 % and a maximum gain of 33.53 %.

We observed that exploiting only the parity of b^+ and b^- our code is on average slightly faster than taking into consideration also the parity of a^+ and a^- . This seems to indicate that the extra operations needed for each section to distinguish between classic and even Toom-2.5 are not always beneficial. More tests on many different hardware and software architectures are needed to have a clearer idea of the general behavior of the code.

6. ACKNOWLEDGEMENTS

A grateful thank goes to Marco Bodrato, for his precious advice and constant encouragement and help in preparing the C code and the paper organization. This work is dedicated to him. Many thanks go also to the anonymous referees, whose observations helped to better this paper.

7. CONCLUSION

An approach to very unbalanced long integer multiplication by means of iterative use of unbalanced Toom-Cook methods has been described. The Toom-2.5 case has been given in some detail, and its performance compared with the code of GMP library. The approach was found to be quite effective in a well-defined region of the product space. Additional unbalanced Toom- k methods are likely to behave effectively as well.

8. REFERENCES

- [1] Anonymous. Personal communication, 2010.
- [2] Marco Bodrato. High degree Toom’n’half for balanced and unbalanced multiplication. In preparation.
- [3] Marco Bodrato. Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In Claude Carlet and Berk Sunar, editors, *WAIFI ’07 proceedings*, volume 4547 of LNCS. Springer, June 2007.
- [4] Marco Bodrato and Alberto Zanoni. Integer and polynomial multiplication: Towards optimal Toom-Cook matrices. In Christopher W. Brown, editor, *Proceedings of the ISSAC 2007 Conference*. ACM press, July 2007.
- [5] Stephen A. Cook. *On the minimum computation time of functions*. PhD thesis, Department of Mathematics, Harvard University, 1966.
- [6] Richard Crandall and Barry Fagin. Discrete weighted transforms and large integer arithmetic. *Mathematics of computation*, 62:305–324, 1994.
- [7] Tom St Denis, Mads Rasmussen, and Greg Rose. Multi-precision math (tommath library documentation). <http://math.libtomcrypt.com/files/tommath.pdf>.
- [8] Torbjörn Granlund. The GNU multiple precision (GMP) library. <http://gmplib.org/>.
- [9] Anatolii Alexeevich Karatsuba and Yuri Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7(7):595–596, 1963.
- [10] Donald E. Knuth. *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.
- [11] Peter L. Montgomery. *An FFT extension of the elliptic curve method of factorization*. PhD thesis, University of California, 1992.
- [12] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7(3–4):281–292, 1971.
- [13] Andrei L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3:714–716, 1963.
- [14] André Weimerskirch and Christof Paar. Generalizations of the Karatsuba algorithm for polynomial multiplication. Technical report, Ruhr-Universität-Bochum, 2003.
- [15] Alberto Zanoni. Some Toom-Cook methods for even long integers. In Daniel Breaz, Nicoleta Breaz, and Dorin Wainberg, editors, *Proceedings of ICTAMI 2009*, pages 807–828. Aeternitas Publishing House 2009.
- [16] Dan Zuras. More on squaring and multiplying large integers. *IEEE Transactions on Computers*, 43(8):899–908, August 1994.

An In-Place Truncated Fourier Transform and Applications to Polynomial Multiplication

David Harvey
Courant Institute of Mathematical Sciences
New York University
New York, New York, U.S.A.
dmharvey@cims.nyu.edu
www.cims.nyu.edu/~harvey/

Daniel S. Roche
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
droche@cs.uwaterloo.ca
www.cs.uwaterloo.ca/~droche/

ABSTRACT

The truncated Fourier transform (TFT) was introduced by van der Hoeven in 2004 as a means of smoothing the “jumps” in running time of the ordinary FFT algorithm that occur at power-of-two input sizes. However, the TFT still introduces these jumps in memory usage. We describe in-place variants of the forward and inverse TFT algorithms, achieving time complexity $O(n \log n)$ with only $O(1)$ auxiliary space. As an application, we extend the second author’s results on space-restricted FFT-based polynomial multiplication to polynomials of arbitrary degree.

Categories and Subject Descriptors

F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—*Computations on polynomials*; G.4 [Mathematical Software]: Algorithm design and analysis, Efficiency; I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms—*Algebraic algorithms, Analysis of algorithms*

General Terms

Algorithms, Performance, Theory

Keywords

Truncated Fourier transform, fast Fourier transform, polynomial multiplication, in-place algorithms

1. INTRODUCTION

1.1 Background

The discrete Fourier transform (DFT) is a linear map that evaluates a given polynomial at powers of a root of unity. An efficient method to compute this transform, known as the Fast Fourier Transform (FFT), was known to Gauss and later rediscovered and extended in the context of digital computing by Cooley and Tukey [3]. This algorithm has since

become one of the most important and useful tools in computer science, most notably in signal processing, scientific computing, and symbolic computation.

In the latter two areas, FFTs have been used extensively to develop asymptotically fast methods for integer and polynomial multiplication [11, 5, 2]. Moreover, numerous other operations on polynomials — including division, evaluation and interpolation, and GCD computation — have been reduced to multiplication, so more efficient multiplication methods have an indirect effect on many areas in computer algebra [6, Ch. 8–11].

The simplest FFT to implement, and often the fastest in practice, is the radix-2 Cooley-Tukey FFT. Because the radix-2 FFT requires the size to be a power of two, the simplest solution for all other sizes is to pad the input polynomials with zeros, resulting in large unwanted “jumps” in the complexity at powers of two.

1.2 The truncated Fourier transform

Numerous approaches have been developed to adapt radix-2 FFTs to arbitrary sized inputs and outputs. The “devil’s convolution” algorithm [4] tackles this issue in the higher-level context of multiplication by breaking the problem into a large power-of-two size, solved with radix-2 FFTs, and a smaller arbitrary size, solved recursively. However, there are still significant jumps in the time and space cost (though not exactly at powers of two).

At the lower-level context of DFTs, it has been known for some time that if only a subset of the output is needed, then the FFT can be truncated or “pruned” to reduce the complexity, essentially by disregarding those parts of the computation tree not contributing to the desired outputs [9, 12]. More recently, van der Hoeven took the crucial step of showing how to invert this process by assuming a subset of input/output coefficients are zero, describing a truncated Fourier transform (TFT) and an *inverse* truncated Fourier transform (ITFT), and showing that this leads to a polynomial multiplication algorithm whose running time varies relatively smoothly in the input size [13, 14].

Specifically, given an input vector of length $n \leq 2^k$, the TFT computes the first n coefficients of the ordinary Fourier transform of length 2^k , and the ITFT computes the inverse of this map. The running time of these algorithms smoothly interpolates the $O(n \log n)$ complexity of the standard radix-2 Cooley–Tukey FFT algorithm. One can therefore deduce an asymptotically fast polynomial multiplication algorithm that avoids the characteristic “jumps” in running time exhibited by traditional FFT-based polynomial multiplication

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

algorithms when the output degree crosses a power-of-two boundary. This observation has been confirmed with practical implementations [14, 8, 7], with the most marked improvements in the multivariate case.

One drawback of van der Hoeven’s algorithms is that while their time complexity varies smoothly with n , their space complexity does not. Both the TFT and ITFT operate in a buffer of length $2^{\lceil \lg n \rceil}$; that is, for inputs of length n , they require auxiliary storage of $2^{\lceil \lg n \rceil} - n + O(1)$ cells to store intermediate results, which can be $\Omega(n)$ in the worst case.

1.3 Summary of results

The main results of this paper are TFT and ITFT algorithms that require only $O(1)$ auxiliary space, while respecting the $O(n \log n)$ time bound.

The new algorithms have their origin in a cache-friendly variant of the TFT and ITFT given by the first author [7], that builds on Bailey’s cache-friendly adaptation of the ordinary FFT [1]. If the transform takes place in a buffer of length $L = 2^\ell$, these algorithms decompose the transform into $L_1 = 2^{\ell_1}$ row transforms of length $L_2 = 2^{\ell_2}$ and L_2 column transforms of length L_1 , where $\ell_1 + \ell_2 = \ell$. Van der Hoeven’s algorithms correspond to the case $L_1 = 2$ and $L_2 = L/2$. To achieve optimal locality, [7] suggests taking $L_i \approx \sqrt{L}$ ($\ell_i \approx \ell/2$). In fact, in this case one already obtains TFT and ITFT algorithms needing only $O(\sqrt{n})$ auxiliary space. At the other extreme we may take $L_1 = L/2$ and $L_2 = 2$, obtaining TFT and ITFT algorithms that use only $O(1)$ space at each recursion level, or $O(\log n)$ auxiliary space altogether. In signal processing language, these may be regarded as decimation-in-time variants of van der Hoeven’s decimation-in-frequency algorithms.

Due to data dependencies in the $O(\log n)$ -space algorithms sketched above, the space usage cannot be reduced further by simply reordering the arithmetic operations. In this paper, we show that with a little extra work, increasing the implied constant in the $O(n \log n)$ running time bound, it is possible to reduce the auxiliary space to only $O(1)$. To make the $O(1)$ space bound totally explicit, we present our TFT and ITFT algorithms (Algorithms 1 and 2) in an iterative fashion, with no recursion. Since we do not have space to store all the necessary roots of unity, we explicitly include steps to compute them on the fly; this is non-trivial because the decimation-in-time approach requires indexing the roots in bit-reversed order.

As an application, we generalize the second author’s space-restricted polynomial multiplication algorithm [10]. Consider a model in which the input polynomials are considered read-only, but the output buffer may be read from and written to multiple times. The second author showed that in such a model, it is possible to multiply polynomials of degree $n = 2^k - 1$ in time $O(n \log n)$ using only $O(1)$ auxiliary space. Using the new in-place ITFT, we generalize this result to polynomials of arbitrary degree.

2. PRELIMINARIES

2.1 Computational model

We work over a ring R containing 2^k -th roots of unity for all (suitably large) k , and in which 2 is not a zero-divisor.

Our memory model is similar to that used in the study of in-place algorithms for sorting and geometric problems, combined with the well-studied notion of algebraic complex-

ity. Specifically, we allow two primitive types in memory: ring elements and pointers. A ring element is any single element of R , and the input to any algorithm will consist of n such elements stored in an array. A pointer can hold a single integer $a \in \mathbb{Z}$ in the range $-cn \leq a \leq cn$ for some fixed constant $c \in \mathbb{N}$. (In our algorithms, we could take $c = 2$.)

We say an algorithm is in-place if it overwrites its input buffer with the output. In this case, any element in this single input/output array may be read from or written to in constant time. Our in-place truncated Fourier transform algorithms (Algorithms 1 and 2) fall under this model.

An out-of-place algorithm uses separate memory locations for input and output. Here, any element from the input array may be read from in constant time (but not overwritten), and any element in the output array may be read from or written to in constant time as well. This will be the situation in our multiplication algorithm (Algorithm 3).

The algorithms also need to store some number of pointers and ring elements not in the input or output arrays, which we define to be the auxiliary storage used by the algorithm. All the algorithms we present will use only $O(1)$ auxiliary storage space.

This model should correspond well with practice, at least when the computations are performed in main memory and the ring R is finite.

2.2 DFT notation

We denote by $\omega_{[k]}$ a primitive 2^k -th root of unity, and we assume that these are chosen compatibly, so that $\omega_{[k+1]}^2 = \omega_{[k]}$ for all $k \geq 0$. Define a sequence of roots $\omega_0, \omega_1, \dots$ by $\omega_s = \omega_{[k]}^{\text{rev}_k s}$, where $k \geq \lceil \lg(s+1) \rceil$ and $\text{rev}_k s$ denotes the length- k bit-reversal of s . Thus we have

$$\begin{aligned} \omega_0 &= \omega_{[0]} (= 1) & \omega_2 &= \omega_{[2]} & \omega_4 &= \omega_{[3]} & \omega_6 &= \omega_{[3]}^3 \\ \omega_1 &= \omega_{[1]} (= -1) & \omega_3 &= \omega_{[2]}^3 & \omega_5 &= \omega_{[3]}^5 & \omega_7 &= \omega_{[3]}^7 \end{aligned}$$

and so on. Note that

$$\omega_{2s+1} = -\omega_{2s} \quad \text{and} \quad \omega_{2s}^2 = \omega_{2s+1}^2 = \omega_s.$$

If $F \in R[x]$ is a polynomial with $\deg F < n$, we write F_s for the coefficient of x^s in F , and we define the Fourier transform $\hat{F}$ by

$$\hat{F}_s = F(\omega_s).$$

In Algorithms 1 and 2 below, we decompose F as

$$F(x) = G(x^2) + xH(x^2),$$

where $\deg G < \lceil n/2 \rceil$ and $\deg H < \lfloor n/2 \rfloor$. Using the properties of ω_s mentioned above, we obtain the “butterfly” relations

$$\begin{aligned} \hat{F}_{2s} &= \hat{G}_s + \omega_{2s} \hat{H}_s, \\ \hat{F}_{2s+1} &= \hat{G}_s - \omega_{2s} \hat{H}_s. \end{aligned} \tag{1}$$

Both the TFT and ITFT algorithm require, at each recursive level, iterating through a set of index-root pairs such as $\{(i, \omega_i), 0 \leq i < n\}$. A traditional, time-efficient approach would be to precompute all powers of $\omega_{[k]}$, store them in reverted-binary order, and then pass through this array with a single pointer. However, this is impossible under the restriction that no auxiliary storage space be used. Instead, we will compute the roots on-the-fly by iterating through the powers of $\omega_{[k]}$ in order, and through the indices i in bit-reversed order. Observe that incrementing an integer

counter through $\text{rev}_k 0, \text{rev}_k 1, \text{rev}_k 2, \dots$ can be done in exactly the same way as incrementing through $0, 1, 2, \dots$, which is possible in-place and in amortized constant time.

3. SPACE-RESTRICTED TFT

In this section we describe an in-place TFT algorithm that uses only $O(1)$ auxiliary space (Algorithm 1). The routine operates on a buffer $X_0, \dots, X_{n-1}$ containing elements of R . It takes as input a root of unity of sufficiently high order and the coefficients $F_0, \dots, F_{n-1}$ of a polynomial $F \in R[x]$, and overwrites these with $\hat{F}_0, \dots, \hat{F}_{n-1}$.

The pattern of the algorithm is recursive, but we avoid recursion by explicitly moving through the recursion tree, avoiding unnecessary space usage. An example tree for $n = 6$ is shown in Figure 1. The node $S = (q, r)$ represents the subarray with offset q and stride 2^r ; the i th element in this subarray is $S_i = X_{q+i \cdot 2^r}$, and the length of the subarray is given by

$$\text{len}(S) = \left\lceil \frac{n-q}{2^r} \right\rceil.$$

The root is $(0, 0)$, corresponding to the entire input array of length n . Each subarray of length 1 corresponds to a leaf node, and we define the predicate $\text{IsLeaf}(S)$ to be true iff $\text{len}(S) = 1$. Each non-leaf node splits into even and odd child nodes. To facilitate the path through the tree, we define

$$\begin{aligned} \text{Even}(q, r) &= (q, r+1), \\ \text{Odd}(q, r) &= (q+2^r, r+1) \end{aligned}$$

if (q, r) is not a leaf,

$$\text{Parent}(q, r) = \begin{cases} (q, r-1), & q < 2^{r-1}, \\ (q-2^{r-1}, r-1), & q \geq 2^{r-1} \end{cases}$$

if (q, r) is not the root, and for any node we define

$$\text{LeftmostLeaf}(S) = \begin{cases} S, & \text{IsLeaf}(S), \\ \text{LeftmostLeaf}(\text{Even}(S)), & \text{otherwise.} \end{cases}$$

We begin with the following lemma.

LEMMA 3.1. *Let N be a node with $\text{len}(N) = \ell$, and let*

$$A(x) = \sum_{0 \leq i < \ell} A_i x^i \in R[x].$$

If $S = \text{LeftmostLeaf}(N)$ and $N_i = A_i$ for $0 \leq i < \ell$ before some iteration of line 3 in Algorithm 1, then after a finite number of steps, we will have $S = N$ and $N_i = \hat{A}_i$ for $0 \leq i < \ell$, before the execution of line 8. No other array entries in X are affected.

PROOF. The proof is by induction on ℓ . If $\ell = 1$, then $\text{IsLeaf}(N)$ is true and $\hat{A}_0 = A_0$ so we are done. So assume $\ell > 1$ and that the lemma holds for all shorter lengths.

Decompose A as $A(x) = G(x^2) + xH(x^2)$. Since $S = \text{LeftmostLeaf}(\text{Even}(N))$ as well, the induction hypothesis guarantees that the even-indexed elements of N , corresponding to the coefficients of G , will be transformed into $\hat{G}$, and we will have $S = \text{Even}(N)$ before line 8. The following lines set $\text{prev} = \text{Even}(N)$ and $S = N$, so that lines 12–16 are executed on the next iteration.

If ℓ is odd, then $(\ell-1)/2 \geq \text{len}(\text{Odd}(N))$, so $\hat{H}_{(\ell-1)/2}$ will not be computed in the odd subtree, and we will not be able

Algorithm 1: InplaceTFT($[X_0, \dots, X_{n-1}]$)

Input: $X_i = F_i$ for $0 \leq i < n$, where $F \in R[x]$,
 $\deg F < n$

Output: $X_i = \hat{F}_i$ for $0 \leq i < n$

```

1  $S \leftarrow \text{LeftmostLeaf}(0, 0)$ 
2  $\text{prev} \leftarrow \text{null}$ 
3 while true do
4    $m \leftarrow \text{len}(S)$ 
5   if  $\text{IsLeaf}(S)$  or  $\text{prev} = \text{Odd}(S)$  then
6     for  $(i, \theta) \in \{(j, \omega_{2j}) : 0 \leq j < \lfloor m/2 \rfloor\}$  do
7        $\begin{bmatrix} S_{2i} \\ S_{2i+1} \end{bmatrix} \leftarrow \begin{bmatrix} S_{2i} + \theta S_{2i+1} \\ S_{2i} - \theta S_{2i+1} \end{bmatrix}$ 
8     if  $S = (0, 0)$  then halt
9      $\text{prev} \leftarrow S$ 
10     $S \leftarrow \text{Parent}(S)$ 
11  else if  $\text{prev} = \text{Even}(S)$  then
12    if  $\text{len}(S) \equiv 1 \pmod 2$  then
13       $v \leftarrow \sum_{i=0}^{(m-3)/2} S_{2i+1} \cdot (\omega_{(m-1)/2})^i$ 
14       $S_{m-1} \leftarrow S_{m-1} + v \cdot \omega_{m-1}$ 
15       $\text{prev} \leftarrow S$ 
16       $S \leftarrow \text{LeftmostLeaf}(\text{Odd}(S))$ 
```

to apply (1) to compute $\hat{A}_{\ell-1} = \hat{G}_{(\ell-1)/2} + \omega_{\ell-1} \hat{H}_{(\ell-1)/2}$. This is why, in this case, we explicitly compute

$$v = H(\omega_{(\ell-1)/2}) = \hat{H}_{(\ell-1)/2}$$

on line 13, and then compute $\hat{A}_{\ell-1}$ directly on line 14, before descending into the odd subtree.

Another application of the induction hypothesis guarantees that we will return to line 8 with $S = \text{Odd}(N)$ after computing $N_{2i+1} = \hat{H}_i$ for $0 \leq i < \lfloor \ell/2 \rfloor$. The following lines set $\text{prev} = \text{Odd}(N)$ and $S = N$, and we arrive at line 6 on the next iteration. The **for** loop thus properly applies the butterfly relations (1) to compute $\hat{A}_i$ for $0 \leq i < 2\lfloor \ell/2 \rfloor$, which completes the proof. $\square$

Now we are ready for the main result of this section.

PROPOSITION 3.2. *Algorithm 1 correctly computes $\hat{F}_i$ for $0 \leq i < n$. It performs $O(n \log n)$ ring and pointer operations, and uses $O(1)$ auxiliary space.*

PROOF. The correctness follows immediately from Lemma 3.1, since we start with $S = \text{LeftmostLeaf}(0, 0)$, which is the first leaf of the whole tree. The space bound is immediate since each variable has constant size.

To verify the time bound, notice that the **while** loop visits each leaf node once and each non-leaf node twice (once with $\text{prev} = \text{Even}(S)$ and once with $\text{prev} = \text{Odd}(S)$). Since always $q < 2^r < 2n$, there are $O(n)$ iterations through the **while** loop, each of which has cost $O(\text{len}(S) + \log n)$. This gives the total cost of $O(n \log n)$. $\square$

4. SPACE-RESTRICTED ITFT

Next we describe an in-place inverse TFT algorithm that uses $O(1)$ auxiliary space (Algorithm 2). It takes as input $\hat{F}_0, \dots, \hat{F}_{n-1}$ for some polynomial $F \in R[x]$, $\deg F < n$, and overwrites the buffer with $F_0, \dots, F_{n-1}$.

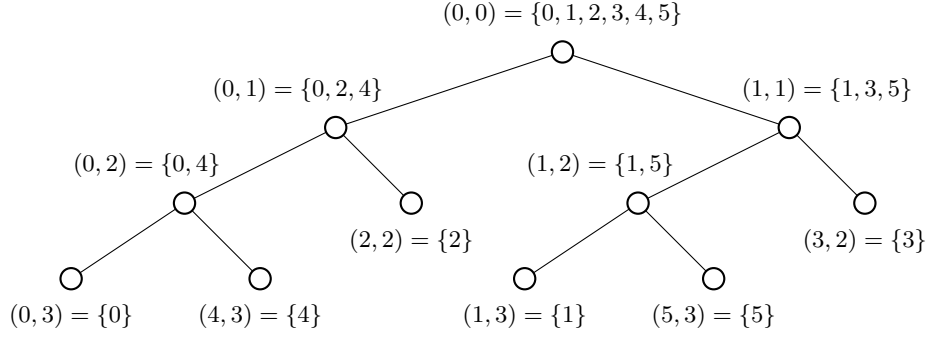


Figure 1: TFT tree for $n = 6$

Algorithm 2: InplaceTFT($[X_0, \dots, X_{n-1}]$)

Input: $X_i = \hat{F}_i$ for $0 \leq i < n$, where $F \in R[x]$,
 $\deg F < n$

Output: $X_i = F_i$ for $0 \leq i < n$

```

1  $S \leftarrow (0, 0)$ 
2 while  $S \neq \text{LeftmostLeaf}(0, 0)$  do
3   if  $\text{IsLeaf}(S)$  then
4      $S \leftarrow \text{Parent}(\text{RightmostParent}(S))$ 
5      $m \leftarrow \text{len}(S)$ 
6     if  $\text{len}(S) \equiv 1 \pmod 2$  then
7        $v \leftarrow \sum_{i=0}^{(m-3)/2} S_{2i+1} \cdot \omega_{(m-1)/2}^i$ 
8        $S_{m-1} \leftarrow S_{m-1} - v \cdot \omega_{m-1}$ 
9      $S \leftarrow \text{Even}(S)$ 
10  else
11     $m \leftarrow \text{len}(S)$ 
12    for  $(i, \theta) \in \{(j, \omega_{2^j}^{-1}) : 0 \leq j < \lfloor m/2 \rfloor\}$  do
13       $\begin{bmatrix} S_{2i} \\ S_{2i+1} \end{bmatrix} \leftarrow \begin{bmatrix} (S_{2i} + S_{2i+1})/2 \\ \theta \cdot (S_{2i} - S_{2i+1})/2 \end{bmatrix}$ 
14     $S \leftarrow \text{Odd}(S)$ 

```

The path of the algorithm is exactly the reverse of Algorithm 1, and we use the same notation as before to move through the tree. We only require one additional function:

$$\text{RightmostParent}(S) = \begin{cases} S, & S = \text{Odd}(\text{Parent}(S)), \\ \text{RightmostParent}(\text{Parent}(S)), & \text{otherwise.} \end{cases}$$

If

$$\text{LeftmostLeaf}(\text{Odd}(N_1)) = N_2,$$

then

$$\text{Parent}(\text{RightmostParent}(N_2)) = N_1,$$

so RightmostParent computes the inverse of the assignment on line 16 in Algorithm 1.

We leave it to the reader to confirm that the structure of the recursion is identical to that of Algorithm 1, but in reverse, from which the following analogues of Lemma 3.1 and Proposition 3.2 follow immediately:

LEMMA 4.1. Let N be a node with $\text{len}(N) = \ell$, and

$$A(x) = \sum_{0 \leq i < \ell} A_i x^i \in R[x].$$

If $S = N$ and $N_i = \hat{A}_i$ for $0 \leq i < \ell$ before some iteration of line 2 in Algorithm 2, then after a finite number of steps, we will have $S = \text{LeftmostLeaf}(N)$ and $N_i = A_i$ for $0 \leq i < \ell$ before some iteration of line 2. No other array entries in X are affected.

PROPOSITION 4.2. Algorithm 2 correctly computes F_i for $0 \leq i < n$. It performs $O(n \log n)$ ring and pointer operations, and uses $O(1)$ auxiliary space.

The fact that our InplaceTFT and InplaceITFT algorithms are essentially reverses of each other is an interesting feature not shared by the original formulations in [13].

5. POLYNOMIAL MULTIPLICATION

We now describe the multiplication algorithm alluded to in the introduction. The strategy is similar to that of [10], with a slightly more complicated “folding” step. The input consists of two polynomials $A, B \in R[x]$ with $\deg A < n$ and $\deg B < m$. The routine is supplied an output buffer X of length $r = n + m - 1$ in which to write the product $C = AB$.

The subroutine FFT has the same interface as InplaceTFT, but is only called for power-of-two length inputs.

PROPOSITION 5.1. Algorithm 3 correctly computes the product $C = AB$, in time $O((m+n) \log(m+n))$ and using $O(1)$ auxiliary space.

PROOF. The main loop terminates since q is strictly increasing. Let N be the number of iterations, and let $q_0 > q_1 > \dots > q_{N-1}$ and $L_0 \geq L_1 \geq \dots \geq L_{N-1}$ be the values of q and L on each iteration. By construction, the intervals $[q_i, q_i + L_i]$ form a partition of $[0, r-1]$, and L_i is the largest power of two such that $q_i + 2L_i \leq r$. Therefore each L can appear at most twice (i.e. if $L_i = L_{i-1}$ then $L_{i+1} < L_i$), $N \leq 2 \lg r$, and we have $L_i \mid q_i$ for each i .

At each iteration, lines 7–8 compute the coefficients of the polynomial $A(\omega_q x) \bmod x^L - 1$, placing the result in $[X_q, \dots, X_{q+L-1}]$. Line 9 then computes $X_{q+i} = A(\omega_q \omega_i)$ for $0 \leq i < L$. Since $L \mid q$ we have $\omega_q \omega_i = \omega_{q+i}$, and so we have actually computed $X_{q+i} = \hat{A}_{q+i}$ for $0 \leq i < L$. The next two lines similarly compute $X_{q+L+i} = \hat{B}_{q+i}$ for $0 \leq i < L$.

Algorithm 3: Space-restricted product

Input: $A, B \in R[x]$, $\deg A < m$, $\deg B < n$
Output: $X_s = C_s$ for $0 \leq s < n + m - 1$, where
 $C = AB$

```
1  $r \leftarrow n + m - 1$ 
2  $q \leftarrow 0$ 
3 while  $q < r - 1$  do
4    $\ell \leftarrow \lfloor \lg(r - q) \rfloor - 1$ 
5    $L \leftarrow 2^\ell$ 
6    $[X_q, X_{q+1}, \dots, X_{q+2L-1}] \leftarrow [0, 0, \dots, 0]$ 
7   for  $0 \leq i < m$  do
8      $X_{q+(i \bmod L)} \leftarrow X_{q+(i \bmod L)} + \omega_q^i A_i$ 
9   FFT( $[X_q, X_{q+1}, \dots, X_{q+L-1}]$ )
10  for  $0 \leq i < n$  do
11     $X_{q+L+(i \bmod L)} \leftarrow X_{q+L+(i \bmod L)} + \omega_q^i B_i$ 
12  FFT( $[X_{q+L}, X_{q+L+1}, \dots, X_{q+2L-1}]$ )
13  for  $0 \leq i < L$  do
14     $X_{q+i} \leftarrow X_{q+i} X_{q+L+i}$ 
15   $q \leftarrow q + L$ 
16  $X_{r-1} \leftarrow A(\omega_{r-1})B(\omega_{r-1})$ 
17 InplaceITFT( $[X_0, \dots, X_{r-1}]$ )
```

(The point of the condition $q + 2L \leq r$ is to ensure that both of these transforms fit into the output buffer.) Lines 13–14 then compute $X_{q+i} = \hat{A}_{q+i} \hat{B}_{q+i} = \hat{C}_{q+i}$ for $0 \leq i < L$.

After line 16 we finally have $X_s = \hat{C}_s$ for all $0 \leq s < r$. (The last product was handled separately since the output buffer does not have room for the two Fourier coefficients.) Line 17 then recovers $C_0, \dots, C_{r-1}$.

We now analyze the time and space complexity. The loops on lines 6, 7, 10 and 13 contribute $O(r)$ operations per iteration, or $O(r \log r)$ in total, since $N = O(\log r)$. The FFT calls contribute $O(L_i \log L_i)$ per iteration, for a total of $O(\sum_i L_i \log L_i) = O(\sum_i L_i \log L) = O(r \log r)$. Line 16 contribute $O(r)$, and line 17 contributes $O(r \log r)$ by Proposition 4.2. The space requirements are immediate also by Proposition 4.2, since the main loop requires only $O(1)$ space. $\square$

6. CONCLUSION

We have demonstrated that forward and inverse radix-2 truncated Fourier transforms can be computed in-place using $O(n \log n)$ time and $O(1)$ auxiliary storage. As a result, polynomials with degrees less than n can be multiplied out-of-place within the same time and space bounds. These results apply to any size n , whenever the underlying ring admits division by 2 and a primitive root of unity of order $2^{\lceil \lg n \rceil}$.

Numerous questions remain open in this direction. First, our in-place TFT and ITFT algorithms avoid using auxiliary space at the cost of some extra arithmetic. So although the asymptotic complexity is still $O(n \log n)$, the implied constant will be greater than for the usual TFT or FFT algorithms. It would be interesting to know whether this extra cost is unavoidable. In any case, the implied constant would need to be reduced as much as possible for the in-place TFT/ITFT to compete with the running time of the original algorithms.

We also have not yet demonstrated an in-place multi-dimensional TFT or ITFT algorithm. In one dimension, the ordinary TFT can hope to gain at most a factor of two over the FFT, but a d -dimensional TFT can be faster than the corresponding FFT by a factor of 2^d , as demonstrated in [8]. An in-place variant along the lines of the algorithms presented in this paper could save a factor of 2^d in both time and memory, with practical consequences for multivariate polynomial arithmetic.

Finally, noticing that our multiplication algorithm, despite using only $O(1)$ auxiliary storage, is still an out-of-place algorithm, we restate an open question of [10]: Is it possible, under any time restrictions, to perform multiplication in-place and using only $O(1)$ auxiliary storage? The answer seems to be no, but a proof is as yet elusive.

7. REFERENCES

- [1] David H. Bailey. FFTs in external or hierarchical memory. *Journal of Supercomputing*, 4:23–35, 1990.
- [2] David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.*, 28(7):693–701, 1991.
- [3] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.
- [4] Richard E. Crandall. *Topics in advanced scientific computation*. Springer-Verlag, New York, 1996.
- [5] Martin Fürer. Faster integer multiplication. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 57–66, New York, NY, USA, 2007. ACM Press.
- [6] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, second edition, 2003.
- [7] David Harvey. A cache-friendly truncated FFT. *Theoret. Comput. Sci.*, 410(27-29):2649–2658, 2009.
- [8] Xin Li, Marc Moreno Maza, and Éric Schost. Fast arithmetic for triangular sets: from theory to practice. *J. Symbolic Comput.*, 44(7):891–907, 2009.
- [9] J. Markel. FFT pruning. *Audio and Electroacoustics, IEEE Transactions on*, 19(4):305–311, Dec 1971.
- [10] Daniel S. Roche. Space- and time-efficient polynomial multiplication. In *ISSAC '09: Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 295–302, New York, NY, USA, 2009. ACM.
- [11] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing (Arch. Elektron. Rechnen)*, 7:281–292, 1971.
- [12] H.V. Sorensen and C.S. Burrus. Efficient computation of the DFT with only a subset of input or output points. *Signal Processing, IEEE Transactions on*, 41(3):1184–1200, Mar 1993.
- [13] Joris van der Hoeven. The truncated Fourier transform and applications. In *ISSAC 2004*, pages 290–296. ACM, New York, 2004.
- [14] Joris van der Hoeven. Notes on the truncated Fourier transform. unpublished, available from <http://www.math.u-psud.fr/~vdhoeven/>, 2005.

Randomized NP-Completeness for p -adic Rational Roots of Sparse Polynomials in One Variable

Martín Avendaño^{*}
TAMU 3368
Mathematics Dept.
College Station, TX 77843-3368, USA
mavendar@yahoo.com.ar

Ashraf Ibrahim^{*}
TAMU 3141
Aerspace Engineering Dept.
College Station, TX 77843-3141, USA
ibrahim@aero.tamu.edu

J. Maurice Rojas^{*}
TAMU 3368
Mathematics Dept.
College Station, TX 77843-3368, USA
rojas@math.tamu.edu

Korben Rusek^{*}
TAMU 3368
Mathematics Dept.
College Station, TX 77843-3368, USA
korben.rusek@gmail.com

ABSTRACT

Relative to the sparse encoding, we show that deciding whether a univariate polynomial has a p -adic rational root can be done in **NP** for most inputs. We also prove a sharper complexity upper bound of **P** for polynomials with suitably generic p -adic Newton polygon. The best previous complexity upper bound was **EXPTIME**. We then prove an unconditional complexity lower bound of **NP**-hardness with respect to randomized reductions, for general univariate polynomials. The best previous lower bound assumed an unproved hypothesis on the distribution of primes in arithmetic progression. We also discuss analogous results over $\mathbb{R}$.

Categories and Subject Descriptors

F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—*Number-theoretic computations*

General Terms

Theory

Keywords

sparse, p -adic, feasibility, NP, arithmetic progression

1. INTRODUCTION

The fields $\mathbb{R}$ and $\mathbb{Q}_p$ (the reals and the p -adic rationals) bear more in common than just completeness with respect to a metric: increasingly, complexity results for one field

have inspired and motivated analogous results in the other (see, e.g., [Coh69, DvdD88] and the pair of works [Kho91] and [Roj04]). We continue this theme by transposing recent algorithmic results for sparse polynomials over the real numbers [BRS09] to the p -adic rationals, sharpening some complexity bounds along the way (see Thm. 1.5 below).

For any commutative ring R with multiplicative identity, let **FEAS_R** — the *R*-feasibility problem (a.k.a. Hilbert’s Tenth Problem over R [DLPvG00]) — denote the problem of deciding whether an input $F \in \bigcup_{k,n \in \mathbb{N}} (\mathbb{Z}[x_1, \dots, x_n])^k$ has a root in R^n . (The underlying input size is clarified in Definition 1.1 below.) Observe that **FEAS_R**, **FEAS_Q**, and $\{\mathbf{FEAS}_{\mathbb{F}_q}\}_q$ a prime power are central problems respectively in algorithmic real algebraic geometry, algorithmic number theory, and cryptography.

For any prime p and $x \in \mathbb{Z}$, recall that the *p*-adic valuation, $\text{ord}_p x$, is the greatest k such that $p^k | x$. We can extend $\text{ord}_p(\cdot)$ to $\mathbb{Q}$ by $\text{ord}_p \frac{a}{b} := \text{ord}_p(a) - \text{ord}_p(b)$ for any $a, b \in \mathbb{Z}$; and we let $|x|_p := p^{-\text{ord}_p x}$ denote the *p*-adic norm. The norm $|\cdot|_p$ defines a natural metric satisfying the ultrametric inequality and $\mathbb{Q}_p$ is, tersely, the completion of $\mathbb{Q}$ with respect to this metric. $|\cdot|_p$ and $\text{ord}_p(\cdot)$ extend naturally to the field of *p*-adic complex numbers $\mathbb{C}_p$, which is the metric completion of the algebraic closure of $\mathbb{Q}_p$ [Rob00, Ch. 3].

We will also need to recall the following containments of complexity classes: $\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{NP} \subseteq \dots \subseteq \mathbf{EXPTIME}$, and the fact that the properness of every inclusion above (save $\mathbf{P} \subseteq \mathbf{EXPTIME}$) is a major open problem [Pap95].

1.1 The Ultrametric Side: Relevance and Results

Algorithmic results over the p -adics are useful in many settings: polynomial-time factoring algorithms over $\mathbb{Q}[x]$ [LLL82], computational complexity [Roj02], studying prime ideals in number fields [Coh94, Ch. 4 & 6], elliptic curve cryptography [Lau04], and the computation of zeta functions [CDV06]. Also, much work has gone into using p -adic methods to algorithmically detect rational points on algebraic plane curves via variations of the **Hasse Principle**¹ (see, e.g., [C-T98, Poo06]). However, our knowledge of the complexity of deciding the existence of solutions for **sparse** polynomial equations over $\mathbb{Q}_p$ is surprisingly coarse: good bounds for the

^{*}Partially supported by Rojas’ NSF CAREER grant DMS-0349309. M.A., J.M.R., and K.R. also partially supported by NSF MCS grant DMS-0915245. J.M.R. and K.R. also partially supported by Sandia National Labs and DOE ASCR grant DE-SC0002505. Sandia is a multiprogram laboratory operated by Sandia Corp., a Lockheed Martin Company, for the US Dept. of Energy’s National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC’10, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

¹If $f \in K[x_1, \dots, x_n]$ is any polynomial and Z_K is its zero set in K^n , then the Hasse Principle is the implication $[Z_{\mathbb{C}} \text{ smooth}, Z_{\mathbb{R}} \neq \emptyset, \text{ and } Z_{\mathbb{Q}_p} \neq \emptyset \text{ for all primes } p] \implies Z_{\mathbb{Q}} \neq \emptyset$. The Hasse Principle is a theorem when $Z_{\mathbb{C}}$ is a quadric hypersurface or a curve of genus zero, but fails in subtle ways already for curves of genus one (see, e.g., [Poo01a]).

number of solutions over $\mathbb{Q}_p$ in one variable weren't even known until the late 1990s [Len99b]. So we focus on precise complexity bounds for polynomials in one variable.

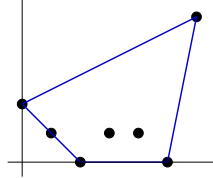
DEFINITION 1.1. Let $f(x) := \sum_{i=1}^m c_i x^{a_i} \in \mathbb{Z}[x]$ satisfy $c_i \neq 0$ for all i , with the a_i pair-wise distinct. We call such an f a **(univariate) m -nomial**. Let us also define $\text{size}(f) := \sum_{i=1}^m \log_2[(2 + |c_i|)(2 + |a_i|)]$ and, for any $F := (f_1, \dots, f_k) \in (\mathbb{Z}[x])^k$, we define $\text{size}(F) := \sum_{i=1}^k \text{size}(f_i)$. Finally, we let $\mathcal{F}_{1,m}$ denote the subset of $\mathbb{Z}[x]$ consisting of polynomials with exactly m monomial terms $\diamond$

The degree, $\deg f$, of a polynomial f can sometimes be exponential in $\text{size}(f)$ for certain families of f , e.g., $d \geq \frac{2^{\text{size}(1+5x^{126}+x^d)}}{2^{16}}$ for all $d \geq 127$. Note also that $\mathbb{Z}[x]$ is the disjoint union $\bigsqcup_{m \geq 0} \mathcal{F}_{1,m}$.

DEFINITION 1.2. Let $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}$ denote the problem of deciding, for an input polynomial system $F \in \bigcup_{k,n \in \mathbb{N}} (\mathbb{Z}[x_1, \dots, x_n])^k$ and an input prime p , whether F has a root in $\mathbb{Q}_p^n$. Also let $\mathbb{P} \subset \mathbb{N}$ denote the set of primes and, when $\mathcal{I}$ is a family of such pairs (F, p) , we let $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathcal{I})$ denote the restriction of $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}$ to inputs in $\mathcal{I}$. The underlying input sizes for $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}$ and $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathcal{I})$ shall then be $\text{size}_p(F) := \text{size}(F) + \log p$ (cf. Definition 1.1). $\diamond$

DEFINITION 1.3. Given any polynomial $f(x) := \sum_{i=1}^m c_i x^{a_i} \in \mathbb{Z}[x]$, we define its **p -adic Newton polygon**, $\text{Newt}_p(f)$, to be the convex hull² of the points $\{(a_i, \text{ord}_p c_i) \mid i \in \{1, \dots, m\}\}$. Also, a face of a polygon $P \subset \mathbb{R}^2$ is called **lower** iff it has an inner normal with positive last coordinate, and the **lower hull** of P is simply the union of all its lower edges. Finally, the polynomial given by summing the terms of f corresponding to points of the form $(a_i, \text{ord}_p c_i)$ in some fixed lower face of $\text{Newt}_p(f)$ is called a (**p -adic**) **lower polynomial**. $\diamond$

EXAMPLE 1.4. For the polynomial $f(x)$ defined as $36 - 8868x + 29305x^2 - 35310x^3 + 18240x^4 - 3646x^5 + 243x^6$, the polygon $\text{Newt}_3(f)$ has exactly 3 lower edges and can easily be verified to resemble the illustration to the right. The polynomial f thus has exactly 2 lower binomials, and 1 lower trinomial. $\diamond$



While there are now randomized algorithms for factoring $f \in \mathbb{Z}[x]$ over $\mathbb{Q}_p[x]$ with expected complexity polynomial in $\text{size}_p(f) + \deg(f)$ [CG00], no such algorithms are known to have complexity polynomial in $\text{size}_p(f)$ alone. Our main theorem below shows that the existence of such an algorithm would imply a complexity collapse nearly as strong as $\mathbf{P} = \mathbf{NP}$. Nevertheless, we obtain new sub-cases of $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathbb{Z}[x] \times \mathbb{P})$ lying in $\mathbf{P}$.

THEOREM 1.5.

1. $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathcal{F}_{1,m} \times \mathbb{P}) \in \mathbf{P}$ for $m \in \{0, 1, 2\}$.
2. For any $(f, p) \in \mathbb{Z}[x] \times \mathbb{P}$ such that f has **no** p -adic lower m -nomials for $m \geq 3$, and p does **not** divide $a_i - a_j$ for any lower binomial with exponents $\{a_i, a_j\}$, we can decide the existence of a root in $\mathbb{Q}_p$ for f in time polynomial in $\text{size}_p(f)$.
3. There is a countable union of algebraic hypersurfaces $\mathcal{E} \subsetneq \mathbb{Z}[x] \times \mathbb{P}$, with natural density 0, such that $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}((\mathbb{Z}[x] \times \mathbb{P}) \setminus \mathcal{E}) \in \mathbf{NP}$. Furthermore, we can decide in $\mathbf{P}$ whether an $f \in \mathcal{F}_{1,3}$ lies in $\mathcal{E}$.

²i.e., smallest convex set containing...

4. If $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathbb{Z}[x] \times \mathbb{P}) \in \mathbf{ZPP}$ then $\mathbf{NP} \subseteq \mathbf{ZPP}$.

5. If the Wagstaff Conjecture is true, then $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathbb{Z}[x] \times \mathbb{P}) \in \mathbf{P} \implies \mathbf{P} = \mathbf{NP}$, i.e., we can strengthen Assertion (4) above.

REMARK 1.6. The **Wagstaff Conjecture**, dating back to 1979 (see, e.g., [BS96, Conj. 8.5.10, pg. 224]), is the assertion that the least prime congruent to $k \bmod N$ is $O(\varphi(N) \log^2 N)$, where $\varphi(N)$ is the number of integers in $\{1, \dots, N\}$ relatively prime to N . Such a bound is significantly stronger than the known implications of the **Generalized Riemann Hypothesis (GRH)**. $\diamond$

While the real analogue of Assertion (1) is easy to prove, $\text{FEAS}_{\mathbb{R}}(\mathcal{F}_{1,3}) \in \mathbf{P}$ was proved only recently [BRS09, Thm. 1.3]. That $\text{FEAS}_{\mathbb{Q}_p}(\mathcal{F}_{1,3}) \in \mathbf{NP}$ for any prime p is surprisingly subtle to prove, having been accomplished by the authors just as this paper went to press [AIRR10].

The intuition behind our algorithmic speed-ups (Assertions (1)–(3)) is that any potential hardness is caused by numerical ill-conditioning, quite similar to the sense long known in numerical linear algebra. Indeed, the classical fact that Newton iteration converges more quickly for a root $\zeta \in \mathbb{C}$ of f with $f'(\zeta)$ having large norm (i.e., a **well-conditioned** root) persists over $\mathbb{Q}_p$. This lies behind the hypotheses of Assertions (2) and (3) (see also Theorem 1.11 below). Note that the hypothesis of Assertion (2) is rather stringent: if one fixes $f \in \mathcal{F}_{1,m}$ with $m \geq 3$ and varies p , then it is easily checked that $\text{Newt}_p(f)$ is a line segment (so the hypothesis fails) for all but finitely many p . On the other hand, the hypothesis for Assertion (3) holds for a significantly large fraction of inputs (see also Proposition 2.13 of Section 2.4).

EXAMPLE 1.7. Let T denote the family of pairs $(f, p) \in \mathbb{Z}[x] \times \mathbb{P}$ with $f(x) = a + bx^{11} + cx^{17} + x^{31}$ and let $T^* := T \setminus \mathcal{E}$. Then there is a sparse 61×61 structured matrix S (cf. Lemma 2.8 in Section 2.3 below) such that $(f, p) \in T^* \iff p \nmid \det S$. So by Theorem 1.5, $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(T^*) \in \mathbf{NP}$, and Proposition 2.13 in Section 3 below tells us that for large coefficients, T^* occupies almost all of T . In particular, letting $T(H)$ (resp. $T^*(H)$) denote those pairs (f, p) in T (resp. T^*) with $|a|, |b|, |c|, p \leq H$, we obtain

$$\frac{\#T^*(H)}{\#T(H)} \geq \left(1 - \frac{244}{2H+1}\right) \left(1 - \frac{1+61 \log(4H) \log H}{H}\right).$$

In particular, one can check via Maple that $(-973 + 21x^{11} - 2x^{17} + x^{31}, p) \in T^*$ for all but 352 primes p . $\diamond$

One subtlety behind Assertion (3) is that $\mathbb{Q}_p$ is uncountable and thus, unlike $\text{FEAS}_{\mathbb{F}_p}$, $\text{FEAS}_{\mathbb{Q}_p}$ does **not** admit an obvious succinct certificate. Indeed, the best previous complexity bound relative to the sparse input size appears to have been $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathbb{Z}[x] \times \mathbb{P}) \in \mathbf{EXPTIME}$ [MW99].³ In particular, $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathcal{F}_{1,4} \times \mathbb{P}) \stackrel{?}{\in} \mathbf{NP}$ and $\text{FEAS}_{\mathbb{R}}(\mathcal{F}_{1,4}) \stackrel{?}{\in} \mathbf{NP}$ are still open questions [BRS09, Sec. 1.2]. A real analogue for Assertion (3) is also unknown at this time.

As for lower bounds, while it is not hard to show that the full problem $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathbb{Z}[x_1, \dots, x_n] \times \mathbb{P})$ is **NP-hard**, the least n making $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathbb{Z}[x_1, \dots, x_n] \times \mathbb{P})$ **NP-hard** appears not to have been known unconditionally. In particular, a weaker version of Assertion (4) was found recently, but only under the truth of an unproved hypothesis on the distribution of primes in arithmetic progression [Roj07a, Main Thm.]. Assertion (4) thus also provides an interesting contrast to earlier work of

³An earlier result claiming $\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathbb{Z}[x] \times \mathbb{P}) \in \mathbf{NP}$ for “most” inputs was announced without proof in [Roj07a, Main Thm.] (see Proposition 1 there).

H. W. Lenstra, Jr. [Len99a], who showed that one can actually find all **low** degree factors of a sparse polynomial (over $\mathbb{Q}[x]$ as opposed to $\mathbb{Q}_p[x]$) in polynomial time. Real analogues to Assertions (4) and (5) are unknown.

1.2 Primes in Random Arithmetic Progressions and a Tropical Trick

The key to proving our lower bound results (Assertions (4) and (5) of Theorem 1.5) is an efficient reduction from a problem discovered to be **NP**-hard by David Alan Plaisted: deciding whether a sparse univariate polynomial vanishes at a complex D^{th} root of unity [Pla84]. Reducing from this problem to its analogue over $\mathbb{Q}_p$ is straightforward, provided $\mathbb{Q}_p^* := \mathbb{Q}_p \setminus \{0\}$ contains a cyclic subgroup of order D where D has sufficiently many distinct prime divisors. We thus need to consider the factorization of $p - 1$, which in turn leads us to primes congruent to 1 modulo certain integers.

While efficiently constructing random primes in **arbitrary** arithmetic progressions remains a famous open problem, we can now at least efficiently build random primes p such that p is moderately sized but $p - 1$ has many prime factors. We use the notation $[j] := \{1, \dots, j\}$ for any $j \in \mathbb{N}$.

THEOREM 1.8. *For any $\delta > 0$, $\varepsilon \in (0, 1/2)$, and $n \in \mathbb{N}$, we can find — within $O\left((n/\varepsilon)^{\frac{3}{2}+\delta} + (n \log(n) + \log(1/\varepsilon))^{7+\delta}\right)$ randomized bit operations — a sequence $P = (p_i)_{i=1}^n$ of consecutive primes and $c \in \mathbb{N}$ such that $p := 1 + c \prod_{i=1}^n p_i$ satisfies $\log p = O(n \log(n) + \log(1/\varepsilon))$ and, with probability $\geq 1 - \varepsilon$, p is prime.*

Theorem 1.8 and its proof are inspired in large part by an algorithm of von zur Gathen, Karpinski, and Shparlinski [vzGKS96, Algorithm following Fact 4.9]. (Theorem 4.10 of [vzGKS96] does not imply Theorem 1.8 above, nor vice-versa.) In particular, they use an intricate random sampling technique to prove that the enumerative analogue of $\text{FEAS}_{\mathbb{F}^{\text{prime powers}}}(\mathbb{Z}[x_1, x_2] \times \mathbb{P})$ is **#P**-hard [vzGKS96, Thm. 4.11].

Our harder upper bound results (Assertions (2) and (3) of Theorem 1.5) will follow in large part from an arithmetic analogue of a key idea from tropical geometry: **toric deformation**. Toric deformation, roughly speaking, means cleverly embedding an algebraic set into a family of algebraic sets 1 dimension higher, in order to invoke combinatorial methods (see, e.g., [EKLO6]). Here, this simply means that we find ways to reduce problems involving general $f \in \mathbb{Z}[x]$ to similar problems involving binomials.

LEMMA 1.9. (See, e.g., [Rob00, Ch. 6, sec. 1.6].) *The number of roots of f in $\mathbb{C}_p$ with valuation v , counting multiplicities, is **exactly** the horizontal length of the lower face of $\text{Newt}_p(f)$ with inner normal $(v, 1)$. ■*

EXAMPLE 1.10. *In Example 1.4 earlier, note that the 3 lower edges have respective horizontal lengths 2, 3, and 1, and inner normals $(1, 1)$, $(0, 1)$, and $(-5, 1)$. Lemma 1.9 then tells us that f has exactly 6 roots in $\mathbb{C}_3$: 2 with 3-adic valuation 1, 3 with 3-adic valuation 0, and 1 with 3-adic valuation -5 . Indeed, one can check that the roots of f are exactly 6, 1, and $\frac{1}{243}$, with respective multiplicities 2, 3, and 1. ♦*

THEOREM 1.11 [AI10, Thm. 4.5] *Suppose $(f, p) \in \mathbb{Z}[x] \times \mathbb{P}$, $(v, 1)$ is an inner normal to a lower edge E of $\text{Newt}_p(f)$, the lower polynomial g corresponding to E is a binomial with exponents $\{a_i, a_j\}$, and p does **not** divide $a_i - a_j$. Then the number of roots $\zeta \in \mathbb{Q}_p$ of f with $\text{ord}_p \zeta = v$ is exactly the number of roots of g in $\mathbb{Q}_p$. ■*

Our main results are proved in Section 3, after the development of some additional theory below.

2. BACKGROUND

Our lower bounds will follow from a chain of reductions involving some basic problems we will review momentarily. We then show how to efficiently construct random primes p such that $p - 1$ has many prime factors in Section 2.2, and then conclude with some quantitative results on resultants in Sections 2.3 and 2.4.

2.1 Roots of Unity and NP-Completeness

Recall that any Boolean expression of one of the following forms:

$$(\diamond) y_i \vee y_j \vee y_k, \neg y_i \vee y_j \vee y_k, \neg y_i \vee \neg y_j \vee y_k, \neg y_i \vee \neg y_j \vee \neg y_k, \\ \text{with } i, j, k \in [3n],$$

is a **3CNFSAT clause**. A **satisfying assignment** for an arbitrary Boolean formula $B(y_1, \dots, y_n)$ is an assignment of values from $\{0, 1\}$ to the variables $y_1, \dots, y_n$ which makes the equality $B(y_1, \dots, y_n) = 1$ true. Let us now refine slightly Plaisted's elegant reduction from **3CNFSAT** to feasibility testing for univariate polynomial systems over the complex numbers [Pla84, Sec. 3, pp. 127–129].

DEFINITION 2.1. *Letting $P := (p_1, \dots, p_n)$ denote any strictly increasing sequence of primes, let us inductively define a semigroup homomorphism $\mathcal{P}_P$ — the **Plaisted morphism with respect to P** — from certain Boolean expressions in the variables $y_1, \dots, y_n$ to $\mathbb{Z}[x]$, as follows:⁴*

- (0) $D_P := \prod_{i=1}^n p_i$, (1) $\mathcal{P}_P(0) := 1$, (2) $\mathcal{P}_P(y_i) := x^{D_P/p_i} - 1$,
- (3) $\mathcal{P}_P(\neg B) := (x^{D_P} - 1) / \mathcal{P}_P(B)$, for any Boolean expression B for which $\mathcal{P}_P(B)$ has already been defined,
- (4) $\mathcal{P}_P(B_1 \vee B_2) := \text{lcm}(\mathcal{P}_P(B_1), \mathcal{P}_P(B_2))$, for any Boolean expressions B_1 and B_2 for which $\mathcal{P}_P(B_1)$ and $\mathcal{P}_P(B_2)$ have already been defined. ♦

LEMMA 2.2. [Pla84, Sec. 3, pp. 127–129] *Suppose $P = (p_i)_{i=1}^n$ is an increasing sequence of primes with $\log(p_k) = O(k^\gamma)$ for some constant γ . Then, for all $n \in \mathbb{N}$ and any clause C of the form $(\diamond)$, we have $\text{size}(\mathcal{P}_P(C))$ polynomial in n^γ . In particular, $\mathcal{P}_P$ can be evaluated at any such C in time polynomial in n . Furthermore, if K is any field possessing D_P distinct D_P^{th} roots of unity, then a **3CNFSAT** instance $B(y) := C_1(y) \wedge \dots \wedge C_k(y)$ has a satisfying assignment iff the univariate polynomial system $F_B := (\mathcal{P}_P(C_1), \dots, \mathcal{P}_P(C_k))$ has a root $\zeta \in K$ satisfying $\zeta^{D_P} = 1$. ■*

Plaisted actually proved the special case $K = \mathbb{C}$ of the above lemma, in slightly different language, in [Pla84]. However, his proof extends verbatim to the more general family of fields detailed above.

2.2 Randomization to Avoid Riemann Hypotheses

The result below allows us to prove Theorem 1.8 and further tailor Plaisted's clever reduction to our purposes. We let $\pi(x)$ denote the number of primes $\leq x$, and let $\pi(x; M, 1)$ denote the number of primes $\leq x$ that are congruent to 1 mod M .

AGP THEOREM. (very special case of [AGP94, Thm. 2.1, pg. 712]) *There exist $x_0 > 0$ and an $\ell \in \mathbb{N}$ such that for each*

⁴Throughout this paper, for Boolean expressions, we will always identify 0 with “False” and 1 with “True”.

$x \geq x_0$, there is a subset $\mathcal{D}(x) \subset \mathbb{N}$ of finite cardinality ℓ with the following property: If $M \in \mathbb{N}$ satisfies $M \leq x^{2/5}$ and $a \notin M$ for all $a \in \mathcal{D}(x)$ then $\pi(x; M, 1) \geq \frac{\pi(x)}{2\varphi(M)}$. ■

For those familiar with [AGP94, Thm. 2.1, pg. 712], the result above follows immediately upon specializing the parameters there as follows:

$$(A, \varepsilon, \delta, y, a) = (49/20, 1/2, 2/245, x, 1)$$

(see also [vzGKS96, Fact 4.9]).

The AGP Theorem enables us to construct random primes from certain arithmetic progressions with high probability. An additional ingredient that will prove useful is the famous **AKS algorithm** for deterministic polynomial-time primality checking [AKS02]. Consider now the following algorithm.

ALGORITHM 2.3.

Input: A constant $\delta > 0$, a failure probability $\varepsilon \in (0, 1/2)$, a positive integer n , and the constants x_0 and ℓ from the AGP Theorem.

Output: An increasing sequence $P = (p_j)_{j=1}^n$ of primes, and $c \in \mathbb{N}$, such that $p := 1 + c \prod_{i=1}^n p_i$ satisfies $\log p = O(n \log(n) + \log(1/\varepsilon))$ and, with probability $1 - \varepsilon$, p is prime. In particular, the output always gives a true declaration as to the primality of p .

Description:

0. Let $L := \lceil 2/\varepsilon \rceil \ell$ and compute the first nL primes $p_1, \dots, p_{nL}$ in increasing order.
1. Define (but do not compute) $M_j := \prod_{k=(j-1)n+1}^{jn} p_k$ for any $j \in \mathbb{N}$. Then compute M_L , M_i for a uniformly random $i \in [L]$, and $x := \max \{x_0, 17, 1 + M_L^{5/2}\}$.
2. Compute $K := \lfloor (x-1)/M_i \rfloor$ and $J := \lceil 2 \log(2/\varepsilon) \log x \rceil$.
3. Pick uniformly random $c \in [K]$ until one either has $p := 1 + cM_i$ prime, or one has J such numbers that are each composite (using primality checks via the AKS algorithm along the way).
4. If a prime p was found then output
“ $1 + c \prod_{j=(i-1)n+1}^{in} p_j$ is a prime that works!”
and stop. Otherwise, stop and output
“I have failed to find a suitable prime. Please forgive me.” ◊

REMARK 2.4. In our algorithm above, it suffices to find integer approximations to the underlying logarithms and square-roots. In particular, we restrict to algorithms that can compute the $\log_2 \mathcal{L}$ most significant bits of $\log \mathcal{L}$, and the $\frac{1}{2} \log_2 \mathcal{L}$ most significant bits of $\sqrt{\mathcal{L}}$, using

$$O((\log \mathcal{L})(\log \log \mathcal{L}) \log \log \log \mathcal{L})$$

bit operations. Arithmetic-Geometric Mean Iteration and (suitably tailored) Newton Iteration are algorithms that respectively satisfy our requirements (see, e.g., [Ber03] for a detailed description). ◊

REMARK 2.5. An anonymous referee suggested that one can employ a faster probabilistic primality test in Step 3 (e.g., [Mor07]), reserving the AKS algorithm solely for so-called **pseudoprimes**. This can likely reduce the complexity bound from Theorem 1.8 slightly. ◊

Proof of Theorem 1.8: It clearly suffices to prove that Algorithm 2.3 is correct, has a success probability that is at least $1 - \varepsilon$, and works within

$$O\left(\left(\frac{n}{\varepsilon}\right)^{\frac{3}{2}+\delta} + (n \log(n) + \log(1/\varepsilon))^{7+\delta}\right)$$

randomized bit operations, for any $\delta > 0$. These assertions are proved directly below. ■

Proving Correctness and the Success Probability Bound for Algorithm 2.3: First observe that $M_1, \dots, M_L$ are relatively prime. So at most ℓ of the M_i will be divisible by elements of $\mathcal{D}(x)$. Note also that $K \geq 1$ and $1 + cM_i \leq 1 + KM_i \leq 1 + ((x-1)/M_i)M_i = x$ for all $i \in [L]$ and $c \in [K]$.

Since $x \geq x_0$ and $x^{2/5} \geq (x-1)^{2/5} \geq (M_i^{5/2})^{2/5} = M_i$ for all $i \in [L]$, the AGP Theorem implies that with probability at least $1 - \frac{\varepsilon}{2}$ (since $i \in [\lceil 2/\varepsilon \rceil \ell]$ is uniformly random), the arithmetic progression $\{1 + M_i, \dots, 1 + KM_i\}$ contains at least $\frac{\pi(x)}{2\varphi(M_i)} \geq \frac{\pi(x)}{2M_i}$ primes. In which case, the proportion of numbers in $\{1 + M_i, \dots, 1 + KM_i\}$ that are prime is $\frac{\pi(x)}{2KM_i} > \frac{\pi(x)}{2+2KM_i} > \frac{x/\log x}{2x} = \frac{1}{2\log x}$, since $\pi(x) > x/\log x$ for all $x \geq 17$ [BS96, Thm. 8.8.1, pg. 233]. So let us now assume that i is fixed and M_i is not divisible by any element of $\mathcal{D}(x)$.

Recalling the inequality $(1 - \frac{1}{t})^{ct} \leq e^{-c}$ (valid for all $c \geq 0$ and $t \geq 1$), we then see that the AGP Theorem implies that the probability of **not** finding a prime of the form $p = 1 + cM_i$ after picking J uniformly random $c \in [K]$ is bounded above by $(1 - \frac{1}{2\log x})^J \leq (1 - \frac{1}{2\log x})^{2\log(2/\varepsilon)\log x} \leq e^{-\log(2/\varepsilon)} = \frac{\varepsilon}{2}$.

In summary, with probability $\geq 1 - \frac{\varepsilon}{2} - \frac{\varepsilon}{2} = 1 - \varepsilon$, Algorithm 2.3 picks an i with M_i not divisible by any element of $\mathcal{D}(x)$ and a c such that $p := 1 + cM_i$ is prime. In particular, we clearly have that

$$\log p = O(\log(1 + KM_i)) = O(n \log(n) + \log(1/\varepsilon)). \quad \blacksquare$$

(Complexity Analysis of Algorithm 2.3): Let $L' := nL$ and, for the remainder of our proof, let p_i denote the i^{th} prime. Since $L' \geq 6$, we have that

$$p_{L'} \leq L'(\log(L') + \log \log L')$$

by [BS96, Thm. 8.8.4, pg. 233]. Recall that the primes in $[\mathcal{L}]$ can be listed simply by deleting all multiples of 2 in $[\mathcal{L}]$, then deleting all multiples of 3 in $[\mathcal{L}]$, and so on until one reaches multiples of $\lfloor \sqrt{\mathcal{L}} \rfloor$. (This is the classic sieve of Eratosthenes.) Recall also that one can multiply an integer in $[\mu]$ and an integer $[\nu]$ within

$O((\log \mu)(\log \log \nu)(\log \log \log \nu) + (\log \nu)(\log \log \mu) \log \log \log \mu)$ bit operations (see, e.g., [BS96, Table 3.1, pg. 43]). So let us define the function $\lambda(a) := (\log \log a) \log \log \log a$.

Step 0: By our preceding observations, it is easily checked that Step 0 takes $O(L'^{3/2} \log^3 L')$ bit operations.

Step 1: This step consists of $n-1$ multiplications of primes with $O(\log L')$ bits (resulting in M_L , which has $O(n \log L')$ bits), multiplication of a small power of M_L by a square root of M_L , division by an integer with $O(n \log L')$ bits, a constant number of additions of integers of comparable size, and the generation of $O(\log L)$ random bits. Employing Remark 2.4 along the way, we thus arrive routinely at an estimate of

$$O(n^2(\log L')\lambda(L') + \log(1/\varepsilon)\lambda(1/\varepsilon))$$

for the total number of bit operations needed for Step 1.

Step 2: Similar to our analysis of Step 1, we see that Step 2 has bit complexity

$$O((n \log(L') + \log(1/\varepsilon))\lambda(n \log L')).$$

Step 3: This is our most costly step: Here, we require

$$O(\log K) = O(n \log(L') + \log(1/\varepsilon))$$

random bits and $J = O(\log x) = O(n \log(L') + \log(1/\varepsilon))$ primality tests on integers with

$$O(\log(1 + cM_i)) = O(n \log(L') + \log(1/\varepsilon))$$

bits. By an improved version of the AKS primality testing algorithm [AKS02, LP05] (which takes $O(N^{6+\delta})$ bit operations to test an N bit integer for primality), Step 3 can then

clearly be done within

$$O((n \log(L') + \log(1/\varepsilon))^{7+\delta})$$

bit operations, and the generation of $O(n \log(L') + \log(1/\varepsilon))$ random bits.

Step 4: This step clearly takes time on the order of the number of output bits, which is just $O(n \log(n) + \log(1/\varepsilon))$ as already observed earlier.

Conclusion: We thus see that Step 0 and Step 3 dominate the complexity of our algorithm, and we are left with an overall randomized complexity bound of

$$\begin{aligned} & O\left(L^{3/2} \log^3(L') + (n \log(L') + \log(1/\varepsilon))^{7+\delta}\right) \\ &= O\left(\left(\frac{n}{\varepsilon}\right)^{3/2} \log^3(n/\varepsilon) + (n \log(n) + \log(1/\varepsilon))^{7+\delta}\right) \\ &= O\left(\left(\frac{n}{\varepsilon}\right)^{\frac{3}{2}+\delta} + (n \log(n) + \log(1/\varepsilon))^{7+\delta}\right) \end{aligned}$$

randomized bit operations. ■

2.3 Transferring from Complex Numbers to p-adics

The proposition below is a folkloric way to reduce systems of univariate polynomial equations to a single polynomial equation, and was already used by Plaisted at the beginning of his proof of Theorem 5.1 in [Pla84].

PROPOSITION 2.6. *Given any $f_1, \dots, f_k \in \mathbb{Z}[x]$ with maximum coefficient absolute value H , let $d := \max_i \deg f_i$ and $\tilde{f}(x) := x^d(f_1(x)f_1(1/x) + \dots + f_k(x)f_k(1/x))$.*

Then $f_1 = \dots = f_k = 0$ has a root on the complex unit circle iff $\tilde{f}$ has a root on the complex unit circle.

Proof: Trivial, upon observing that $f_i(x)f_i(1/x) = |f_i(x)|^2$ for all $i \in [k]$ and any $x \in \mathbb{C}$ with $|x| = 1$. ■

By introducing the classical univariate resultant we will be able to derive the explicit quantitative bounds we need.

DEFINITION 2.7. (See, e.g., [GKZ94, Ch. 12, Sec. 1, pp. 397–402].) Suppose $f(x) = a_0 + \dots + a_d x^d$ and $g(x) = b_0 + \dots + b_{d'} x^{d'}$ are polynomials with indeterminate coefficients. We define their **Sylvester matrix** to be the $(d+d') \times (d+d')$ matrix

$$S_{(d,d')}(f,g) := \begin{bmatrix} a_0 & \dots & a_d & 0 & \dots & 0 \\ & \ddots & & & \ddots & \\ 0 & \dots & 0 & a_0 & \dots & a_d \\ b_0 & \dots & b_{d'} & 0 & \dots & 0 \\ & \ddots & & & \ddots & \\ 0 & \dots & 0 & b_0 & \dots & b_{d'} \end{bmatrix} \left. \begin{array}{l} \left. \begin{array}{l} \vdots \\ \vdots \end{array} \right\} d' \text{ rows} \\ \left. \begin{array}{l} \vdots \\ \vdots \end{array} \right\} d \text{ rows} \end{array} \right\}$$

and their **Sylvester resultant** to be $\mathcal{R}_{(d,d')}(f,g) := \det S_{(d,d')}(f,g)$. ◇

LEMMA 2.8. *Following the notation of Definition 2.7, assume $f, g \in K[x]$ for some field K , and that a_d and $b_{d'}$ are not both 0. Then $f = g = 0$ has a root in the algebraic closure of K iff $\mathcal{R}_{(d,d')}(f,g) = 0$. More generally, we have $\mathcal{R}_{(d,d')}(f,g) = a_d^{d'} \prod_{f(\zeta)=0} g(\zeta)$ where the product counts multiplicity.*

Finally, if we assume further that f and g have complex coefficients of absolute value $\leq H$, and f (resp. g) has exactly m (resp. m') monomial terms, then $|\mathcal{R}_{(d,d')}(f,g)| \leq m^{d'/2} m'^{d/2} H^{d+d'}$. ■

The first 2 assertions are classical (see, e.g., [GKZ94, Ch. 12, Sec. 1, pp. 397–402] and [RS02, pg. 9]). The last assertion follows easily from Hadamard's Inequality (see, e.g., [Mig82, Thm. 1, pg. 259]).

A simple consequence of our last lemma is that vanishing at a D^{th} root of unity is algebraically the same thing over $\mathbb{C}$ or $\mathbb{Q}_p$, provided p lies in the right arithmetic progression.

LEMMA 2.9. *Suppose $D \in \mathbb{N}$, $f \in \mathbb{Z}[x]$, and p is any prime congruent to 1 mod D . Then f vanishes at a complex D^{th}*

root of unity $\iff f$ vanishes at a D^{th} root of unity in $\mathbb{Q}_p$.

REMARK 2.10 *Note that $x^2 + x + 1$ vanishes at a 3^{rd} root of unity in $\mathbb{C}$, but has no roots at all in $\mathbb{F}_5$ or $\mathbb{Q}_5$. So our congruence assumption on p is necessary. ◇*

Proof of Lemma 2.9: First note that by our assumption on p , $\mathbb{Q}_p$ has D distinct D^{th} roots of unity: This follows easily from Hensel's Lemma (see, e.g., [Rob00, Pg. 48]) and $\mathbb{F}_p$ having D distinct D^{th} roots of unity. Since $\mathbb{Z} \hookrightarrow \mathbb{Q}_p$ and $\mathbb{Q}_p$ contains all D^{th} roots of unity by construction, the equivalence then follows directly from Lemma 2.8. ■

2.4 Good Inputs and Bad Trinomials

DEFINITION 2.11. *For any field K , write any $f \in K[x]$ as $f(x) = \sum_{i=1}^m c_i x^{a_i}$ with $0 \leq a_1 < \dots < a_m$. Letting $\mathcal{A} = \{a_1, \dots, a_m\}$, and following the notation of Lemma 2.9, we then define the **$\mathcal{A}$ -discriminant** of f , $\Delta_{\mathcal{A}}(f)$, to be*

$$\mathcal{R}_{(\bar{a}_m, \bar{a}_m - \bar{a}_2)} \left(\tilde{f}, \frac{\partial \tilde{f}}{\partial x} / x^{\bar{a}_2 - 1} \right) / c_m^{\bar{a}_m - \bar{a}_m - 1},$$

*where $\bar{a}_i := (a_i - a_1)/g$ for all i , $\tilde{f}(x) := \sum_{i=1}^m c_i x^{\bar{a}_i}$, and $g := \gcd(a_2 - a_1, \dots, a_m - a_1)$ (see also [GKZ94, Ch. 12, pp. 403–408]). Finally, if $c_i \neq 0$ for all i , then we call $\text{Supp}(f) := \{a_1, \dots, a_m\}$ the **support** of f . ◇*

REMARK 2.12 *Note that when $\mathcal{A} = \{0, \dots, d\}$ we have $\Delta_{\mathcal{A}}(f) = \mathcal{R}_{(d,d-1)}(f, f')/c_d$, i.e., for dense polynomials, the $\mathcal{A}$ -discriminant agrees with the classical discriminant ◇*

Let us now clarify our statement about natural density 0 from Assertion (4) of Theorem 1.5: First, let $(\mathbb{Z} \times (\mathbb{N} \cup \{0\}))^\infty$ denote the set of all infinite sequences of pairs $((c_i, a_i))_{i=1}^\infty$ with $c_i = a_i = 0$ for i sufficiently large. Note then that $\mathbb{Z}[x]$ admits a natural embedding into $(\mathbb{Z} \times (\mathbb{N} \cup \{0\}))^\infty$ by considering coefficient-exponent pairs in order of increasing exponents, e.g., $a + bx^{99} + x^{2001} \mapsto ((a, 0), (b, 99), (1, 2001), (0, 0), (0, 0), \dots)$. Then natural density for a set of pairs $\mathcal{I} \subseteq \mathbb{Z}[x] \times \mathbb{P}$ simply means the corresponding natural density within $(\mathbb{Z} \times (\mathbb{N} \cup \{0\}))^\infty \times \mathbb{P}$. In particular, our claim of natural density 0 can be made explicit as follows.

PROPOSITION 2.13. *For any subset $\mathcal{A} = \{a_1, \dots, a_m\} \subset \mathbb{N} \cup \{0\}$ with $0 = a_1 < \dots < a_m$, let $T_{\mathcal{A}}$ denote the family of pairs $(f, p) \in \mathbb{Z}[x] \times \mathbb{P}$ with $f(x) = \sum_{i=1}^m c_i x^{a_i}$ and let $T_{\mathcal{A}}^*$ denote the subset of $T_{\mathcal{A}}$ consisting of those pairs (f, p) with $p \nmid \Delta_{\mathcal{A}}(f)$. Also let $T_{\mathcal{A}}(H)$ (resp. $T_{\mathcal{A}}^*(H)$) denote those pairs (f, p) in $T_{\mathcal{A}}$ (resp. $T_{\mathcal{A}}^*$) where $|c_i| \leq H$ for all $i \in [m]$ and $p \leq H$. Finally, let $d := a_m / \gcd(a_2, \dots, a_m)$. Then for all $H \geq 17$ we have*

$$\frac{\#T_{\mathcal{A}}^*(H)}{\#T_{\mathcal{A}}(H)} \geq \left(1 - \frac{(2d-1)m}{2H+1}\right) \left(1 - \frac{1+(2d-1)\log(mH)\log(H)}{H}\right).$$

Note that each $T_{\mathcal{A}}^*(H)$ is the complement of a union of hypersurfaces (one for each mod p reduction of $\Delta_{\mathcal{A}}(f)$) in a “brick” in $\mathbb{Z}^m \times \mathbb{P}$. We will see in the proof of Assertion (3) of Theorem 1.5 that the exceptional set $\mathcal{E}$ is then merely the complement of the union $\bigcup_{\mathcal{A}} T_{\mathcal{A}}^*$ as $\mathcal{A}$ ranges over all finite subsets of $\mathbb{N} \cup \{0\}$. Our proposition above is proved in Section 3.2.

Before proving our main results, let us make some final observations about the roots of trinomials.

COROLLARY 2.14. *Suppose $f(x) = c_1 + c_2 x^{a_2} + c_3 x^{a_3} \in \mathcal{F}_{1,3}$, $\mathcal{A} := \{0, a_2, a_3\}$, $0 < a_2 < a_3$, $a_3 \geq 3$, and $\gcd(a_2, a_3) = 1$. Then: (0) $\Delta_{\mathcal{A}}(f) = (a_3 - a_2)^{a_3 - a_2} a_2^{a_2} c_2^{a_3} - (-a_3)^{a_3} c_1^{a_3 - a_2} c_3^{a_2}$. (1) $\Delta_{\mathcal{A}}(f) \neq 0 \iff f$ has no degenerate roots. In which case, we also have $\Delta_{\mathcal{A}}(f) = \frac{(-1)^{a_3} c_3^{a_2 - 1}}{c_1^{a_2 - 1}} \prod_{f(\zeta)=0} f'(\zeta)$.*

(2) Deciding whether f has a degenerate root in $\mathbb{C}_p$ can be done in time polynomial in $\text{size}_p(f)$.

Proof: (0): [GKZ94, Prop. 1.8, pg. 274]. ■

(1): The first assertion follows directly from Definition 2.11 and the vanishing criterion for $\text{Res}_{(a_3, a_3 - a_2)}$ from Lemma 2.8. To prove the second assertion, observe that the product formula from Lemma 2.8 implies that

$$\begin{aligned} \Delta_{\mathcal{A}}(f) &= c_3^{a_3 - a_2} \left(\prod_{f(\zeta)=0} \frac{f'(\zeta)}{\zeta^{a_2 - 1}} \right) / c_3^{a_3 - a_2} \\ &= (-1)^{a_3} \left(\prod_{f(\zeta)=0} f'(\zeta) \right) / (c_1/c_3)^{a_2 - 1}. \quad \blacksquare \end{aligned}$$

(2): From Assertion (1) it suffices to detect the vanishing of $\Delta_{\mathcal{A}}(f)$. However, while Assertion (0) implies that one can evaluate $\Delta_{\mathcal{A}}(f)$ with a small number of arithmetic operations, the bit-size of $\Delta_{\mathcal{A}}(f)$ can be quite large. Nevertheless, we can decide within time polynomial in $\text{size}(f)$ whether these particular $\Delta_{\mathcal{A}}(f)$ vanish for integer c_i via **gcd-free bases** (see, e.g., [BRS09, Sec. 2.4]). ■

We will also need a concept that is essentially the opposite of a degenerate root: Given any $f \in \mathbb{Z}[x]$, we call a $\zeta_0 \in \mathbb{Z}/p^\ell \mathbb{Z}$ an **approximate root** iff $f(\zeta_0) = 0 \pmod{p^\ell}$ and $\text{ord}_p f'(\zeta_0) < \ell/2$, i.e., ζ_0 satisfies the hypotheses of Hensel's Lemma (see, e.g., [Rob00, Pg. 48]), and thus ζ_0 can be lifted to a **p -adic integral root** ζ of f . The terminology “approximate root” is meant to be reminiscent of an Archimedean analogue guaranteeing that $\zeta_0 \in \mathbb{C}$ converge quadratically to a true (non-degenerate) complex root of f (see, e.g., [Sma86]).

We call any $\text{Newt}_p(f)$ such that f has no lower m -nomials with $m \geq 3$ **generic**. Finally, if $p|(a_i - a_j)$ with $\{a_i, a_j\}$ the exponents of some lower binomial of f then we call $\text{Newt}_p(f)$ **ramified**.

3. PROVING OUR MAIN RESULTS

3.1 The Proof of Theorem 1.5

Assertion (1) ($\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathcal{F}_{1,m} \times \mathbb{P}) \in \mathbb{P}$ for $m \leq 2$):

First note that the case $m \leq 1$ is trivial: such a univariate m -nomial has no roots in $\mathbb{Q}_p$ iff it is a nonzero constant. So let us now assume $m = 2$.

We can easily reduce to the special case $f(x) := x^d - \alpha$ with $\alpha \in \mathbb{Q}^*$, since we can divide any input by a suitable monomial term, and arithmetic over $\mathbb{Q}$ is doable in polynomial time. Clearly then, any p -adic root ζ of $x^d - \alpha$ satisfies $d \text{ord}_p \zeta = \text{ord}_p \alpha$. Since we can compute $\text{ord}_p \alpha$ and reductions of integers mod d in polynomial-time [BS96, Ch. 5], we can then assume that $d | \text{ord}_p \alpha$ (for otherwise, f would have no roots over $\mathbb{Q}_p$). Replacing $f(x)$ by $p^{-\text{ord}_p \alpha} f(p^{\text{ord}_p \alpha / d} x)$, we can assume further that $\text{ord}_p \alpha = \text{ord}_p \zeta = 0$. In particular, if $\text{ord}_p \alpha$ was initially a nonzero multiple of d , then $\log \alpha \geq d \log_2 p$. So $\text{size}(f) \geq d$ and our rescaling at worst doubles $\text{size}(f)$.

Letting $k := \text{ord}_p d$, note that $f'(x) = dx^{d-1}$ and thus $\text{ord}_p f'(\zeta) = \text{ord}_p d + (d-1)\text{ord}_p \zeta = k$. So by Hensel's Lemma it suffices to decide whether the mod p^ℓ reduction of f has a root in $(\mathbb{Z}/p^\ell \mathbb{Z})^*$, for $\ell = 1 + 2k$. Note in particular that $\text{size}(p^\ell) = O(\log(p) \text{ord}_p d) = O(\log(p) \log(d) / \log p) = O(\log d)$ which is linear in our notion of input size. Since the equation $x^d = \alpha$ can be solved in any cyclic group via a fast exponentiation, we can then clearly decide whether $x^d - \alpha$ has a root in $(\mathbb{Z}/p^\ell \mathbb{Z})^*$ within $\mathbb{P}$, provided $p^\ell \notin \{8, 16, 32, \dots\}$. This is because of the classical structure theorem for the multiplicative group of $\mathbb{Z}/p^\ell \mathbb{Z}$ (see, e.g., [BS96, Thm. 5.7.2 & Thm. 5.6.2, pg. 109]).

To dispose of the remaining cases $p^\ell \in \{8, 16, 32, \dots\}$, first recall that the multiplicative group of $\mathbb{Z}/2^\ell$ is exactly

$$\{\pm 1, \pm 5, \pm 5^2, \pm 5^3, \dots, \pm 5^{2^{\ell-2}-1} \pmod{2^\ell}\}$$

(see, e.g., [BS96, Thm. 5.7.2 & Thm. 5.6.2, pg. 109]). So we can replace d by its reduction mod $2^{\ell-2}$, since every element of $(\mathbb{Z}/2^\ell \mathbb{Z})^*$ has order dividing $2^{\ell-2}$, and this reduction can certainly be computed in polynomial-time. Let us then write $d = 2^h d'$ where $2 \nmid d'$ and $h \in \{0, \dots, \ell-3\}$, and compute $d'' := 1/d' \pmod{2^{\ell-2}}$. Clearly then, $x^d - \alpha$ has a root in $(\mathbb{Z}/2^\ell \mathbb{Z})^*$ iff $x^{2^h} - \alpha'$ has a root in $(\mathbb{Z}/2^\ell \mathbb{Z})^*$, where $\alpha' := \alpha^{d''}$ (since exponentiation by any odd power is an automorphism of $(\mathbb{Z}/2^\ell \mathbb{Z})^*$). Note also that α' , d' , and d'' can be computed in polynomial time via recursive squaring and standard modular arithmetic, and $h \leq \log_2 d$.

Since $x^{2^h} - \alpha'$ always has a root in $(\mathbb{Z}/2^\ell \mathbb{Z})^*$ when $h=0$, we can then restrict our root search to the cyclic subgroup $\{1, 5^2, 5^4, 5^6, \dots, 5^{2^{\ell-2}-2}\}$ when $h \geq 1$ and α' is a square (since there can be no roots when $h \geq 1$ and α' is not a square). Furthermore, we see that $x^{2^h} - \alpha'$ can have no roots in $(\mathbb{Z}/2^\ell \mathbb{Z})^*$ if $\text{ord}_2 \alpha'$ is odd. So, by rescaling x , we can assume further that $\text{ord}_2 \alpha' = 0$, and thus that α' is odd. Now an odd α' is a square in $(\mathbb{Z}/2^\ell \mathbb{Z})^*$ iff $\alpha' \equiv 1 \pmod{8}$ [BS96, Ex. 38, pg. 192], and this can clearly be checked in $\mathbb{P}$. So we can at last decide the existence of a root in $\mathbb{Q}_2$ for $x^d - \alpha$ in $\mathbb{P}$: Simply use fast exponentiation to solve the equation $x^{2^h} = \alpha'$ over the cyclic subgroup $\{1, 5^2, 5^4, 5^6, \dots, 5^{2^{\ell-2}-2}\}$ of $(\mathbb{Z}/2^\ell \mathbb{Z})^*$ [BS96, Thm. 5.7.2 & Thm. 5.6.2, pg. 109]. ■

Assertion (2) ($\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathbb{Z}[x] \times \mathbb{P}) \in \mathbb{P}$ for generic, unramified $\text{Newt}_p(f)$):

Assertion (2) follows directly from Theorem 1.11, since we can apply the $m = 2$ case of Assertion (1) to the resulting lower binomials. In particular, note that the number of lower binomials of f is no more than the number of monomial terms of f , which is in turn bounded above by $\text{size}(f)$, so the complexity is indeed $\mathbb{P}$. ■

Assertion (3) ($\text{FEAS}_{\mathbb{Q}_{\text{primes}}}(\mathbb{Z}[x] \times \mathbb{P}) \in \text{NP}$ usually):

Let us first observe that it suffices to prove that, for most inputs, we can detect roots in $\mathbb{Z}_p$ in NP . This is because $x \in \mathbb{Q}_p \setminus \mathbb{Z}_p \iff \frac{1}{x} \in p\mathbb{Z}_p$, so letting $f^*(x) := x^{\deg f} f(1/x)$ denote the **reciprocal polynomial** of f , the set of p -adic rational roots of f is simply the union of the p -adic integer roots of f and the reciprocals of the p -adic integer roots of f^* . We may also assume that f is not divisible by x .

Note also that we can find the p -parts of the c_i in polynomial-time via gcd-free bases [BRS09, Sec. 2.4] and thus compute $\text{Newt}_p(f)$ in time polynomial in $\text{size}_p(f)$ (via standard convex hull algorithms, e.g., [Ede87]). Since $\text{ord}_p c_i \leq \log_p c_i \leq \text{size}(c_i)$, note also that that every root $\zeta \in \mathbb{C}_p$ of f satisfies $|\text{ord}_p \zeta| \leq 2 \max_i \text{size}(c_i) \leq 2 \text{size}(f) < 2 \text{size}_p(f)$.

Since $\text{ord}_p(\mathbb{Z}_p) = \mathbb{N} \cup \{0\}$, we can clearly assume that $\text{Newt}_p(f)$ has an edge with non-positive integral slope, for otherwise f would have no roots in $\mathbb{Z}_p$. Letting $g(x) := f'(x)/x^{a_1-1}$, and $\zeta \in \mathbb{Z}_p$ be any p -adic integer root of f , note then that

$$(\star) \quad \text{ord}_p f'(\zeta) = (a_1 - 1) \text{ord}_p(\zeta) + \text{ord}_p g(\zeta).$$

Note also that $\Delta_{\mathcal{A}}(f) = \text{Res}_{(a_m, a_m - a_1)}(f, g)$ so if $p \nmid \Delta_{\mathcal{A}}(f)$ then f and g have no common roots in the algebraic closure of $\mathbb{F}_p$, by Lemma 2.8. In particular, $p \nmid \Delta_{\mathcal{A}}(f) \implies g(\zeta) \not\equiv 0 \pmod{p}$; and thus $p \nmid \Delta_{\mathcal{A}}(f, g) \implies \text{ord}_p f'(\zeta) = (a_1 - 1) \text{ord}_p(\zeta)$. Furthermore, by the convexity of the lower

hull of $\text{Newt}_p(f)$, it is clear that $\text{ord}_p(\zeta) \leq \frac{\text{ord}_p c_0 - \text{ord}_p c_i}{a_i}$ where $(a_i, \text{ord}_p c_i)$ is the rightmost vertex of the lower edge of $\text{Newt}_p(f)$ with least (non-positive and integral) slope. Clearly then, $\text{ord}_p(\zeta) \leq \frac{2 \max_i \log_p |c_i|}{a_1}$. So $p \nmid \Delta_{\mathcal{A}}(f) \implies \text{ord}_p f'(\zeta) \leq 2\text{size}(f)$, thanks to $(\star)$.

Our fraction of inputs admitting a succinct certificate will then correspond precisely to those (f, p) such that $p \nmid \Delta_{\mathcal{A}}(f)$. In particular, let us define $\mathcal{E}$ to be the union of all pairs (f, p) such that $p \mid \Delta_{\mathcal{A}}(f)$, as $\mathcal{A}$ ranges over all finite subsets of $\mathbb{N} \cup \{0\}$. It is then easily checked that $\mathcal{E}$ is a countable union of hypersurfaces.

Now fix $\ell = 4\text{size}(f) + 1$. Clearly then, by Hensel's Lemma, for any $(f, p) \in (\mathbb{Z}[x] \times \mathbb{P}) \setminus \mathcal{E}$, f has a root $\zeta \in \mathbb{Z}_p \iff f$ has a root $\zeta_0 \in \mathbb{Z}/p^\ell \mathbb{Z}$. Since $\log(p^\ell) = O(\text{size}(f) \log p) = O(\text{size}_p(f)^2)$, and since arithmetic in $\mathbb{Z}/p^\ell \mathbb{Z}$ can be done in time polynomial in $\log(p^\ell)$ [BS96, Ch. 5], we have thus at last found our desired certificate: an approximate root $\zeta_0 \in (\mathbb{Z}/p^\ell \mathbb{Z})^*$ of f with $\ell = 4\text{size}(f) + 1$. ■

Assertion (4) ($\text{FEAS}_{\mathbb{Q}\text{primes}}(\mathbb{Z}[x] \times \mathbb{P})$ is NP-hard under ZPP-reductions):

We will prove a (ZPP) randomized polynomial-time reduction from 3CNFSAT to $\text{FEAS}_{\mathbb{Q}\text{primes}}(\mathbb{Z}[x] \times \mathbb{P})$, making use of the intermediate input families $\{(\mathbb{Z}[x])^k \mid k \in \mathbb{N}\} \times \mathbb{P}$ and $\mathbb{Z}[x] \times \{x^D - 1 \mid D \in \mathbb{N}\} \times \mathbb{P}$ along the way.

Toward this end, suppose $B(y) := C_1(y) \wedge \dots \wedge C_k(y)$ is any 3CNFSAT instance. The polynomial system $(\mathcal{P}_P(C_1), \dots, \mathcal{P}_P(C_k))$, for P the first n primes (employing Lemma 2.2), then clearly yields $\text{FEAS}_{\mathbb{C}}(\{(\mathbb{Z}[x])^k \mid k \in \mathbb{N}\}) \in \mathbf{P} \implies \mathbf{P} = \mathbf{NP}$. Composing this reduction with Proposition 2.6, we then immediately obtain $\text{FEAS}_{\mathbb{C}}(\mathbb{Z}[x] \times \{x^D - 1 \mid D \in \mathbb{N}\}) \in \mathbf{P} \implies \mathbf{P} = \mathbf{NP}$.

We now need only find a means of transferring from $\mathbb{C}$ to $\mathbb{Q}_p$. This we do by preceding our reductions above by a judicious (possibly new) choice of P : by applying Theorem 1.8 with $\varepsilon = 1/3$ (cf. Lemma 2.9) we immediately obtain the implication $\text{FEAS}_{\mathbb{Q}\text{primes}}((\mathbb{Z}[x] \times \{x^D - 1 \mid D \in \mathbb{N}\}) \times \mathbb{P}) \in \mathbf{ZPP} \implies \mathbf{NP} \subseteq \mathbf{ZPP}$.

To conclude, observe that any root $(x, y) \in \mathbb{Q}_p^2 \setminus \{(0, 0)\}$ of the quadratic form $x^2 - py^2$ must satisfy $2\text{ord}_p x = 1 + 2\text{ord}_p y$ (an impossibility). So the only p -adic rational root of $x^2 - py^2$ is $(0, 0)$ and we easily obtain a polynomial-time reduction from $\text{FEAS}_{\mathbb{Q}\text{primes}}((\mathbb{Z}[x] \times \{x^D - 1 \mid D \in \mathbb{N}\}) \times \mathbb{P})$ to $\text{FEAS}_{\mathbb{Q}\text{primes}}(\mathbb{Z}[x] \times \mathbb{P})$: simply map any instance $(f(x), x^D - 1, p)$ of the former problem to $(f(x)^2 - (x^D - 1)^2 p, p)$. So we are done. ■

Assertion (5) ($\text{FEAS}_{\mathbb{Q}\text{primes}}(\mathbb{Z}[x] \times \mathbb{P})$ is NP-hard, assuming Wagstaff's Conjecture):

If we also have the truth of the Wagstaff Conjecture then we simply repeat our last proof, replacing our AGP Theorem-based algorithm with a simple brute-force search. More precisely, letting $D := 2 \cdot 3 \cdots p_n$, we simply test the integers $1 + kD$ for primality, starting with $k = 1$ until one finds a prime. If Wagstaff's Conjecture is true then we need not proceed any farther than $k = O\left(\frac{\varphi(D)}{D} \log^2 D\right)$. (Note that $1 \leq \frac{\varphi(D)}{D} < D$ for all $D \geq 2$.) Using the AKS algorithm, this brute-force search clearly has (deterministic) complexity polynomial in $\log D$ which in turn is polynomial in n . ■

3.2 The Proof of Proposition 2.13

By the Schwartz-Zippel Lemma [Sch80], $\Delta_{\mathcal{A}}(f)$ vanishes for at most $(2d - 1)m(2H + 1)^{m-1}$ selections of coefficients

from $\{-H, \dots, H\}$. In other words, $\Delta_{\mathcal{A}}(f) = 0$ for a fraction of at most $\frac{(2d-1)m}{2H+1}$ of the pairs $(f, p) \in T_{\mathcal{A}}(H)$.

Clearly, a pair $(f, p) \in T_{\mathcal{A}}(H)$ for which $p \nmid \Delta_{\mathcal{A}}(f)$ must satisfy $\Delta_{\mathcal{A}}(f) \neq 0$. We have just shown that the fraction of $T_{\mathcal{A}}(H)$ satisfying the last condition is at least $1 - \frac{(2d-1)m}{2H+1}$. Once we show that, amongst these pairs, at least

$$1 - \frac{1 + (2d-1)\log(mH)}{H/\log H}$$

of them actually satisfy $p \nmid \Delta_{\mathcal{A}}(f)$, then we will be done.

To prove the last lower bound, note that $\Delta_{\mathcal{A}}(f)$ has degree at most $2d - 1$ in the coefficients of f by Lemma 2.8. Also, for any fixed $f \in T_{\mathcal{A}}(H)$, $\Delta_{\mathcal{A}}(f)$ is an integer as well, and is thus divisible by no more than $1 + (2d - 1)\log(mH)$ primes if $\Delta_{\mathcal{A}}(f) \neq 0$. (This follows from Lemma 2.8 again, and the elementary fact that an integer N has no more than $1 + \log N$ distinct prime factors.) Recalling that $\pi(x) > x/\log x$ for all $x \geq 17$ [BS96, Thm. 8.8.1, pg. 233], we thus obtain that the fraction of primes $\leq H$ dividing a nonzero $\Delta_{\mathcal{A}}(f)$ is bounded above by $\frac{1 + (2d-1)\log(mH)}{H/\log H}$. ■

Acknowledgements

The authors would like to thank David Alan Plaisted for his kind encouragement, and Eric Bach, Sidney W. Graham, and Igor Shparlinski for many helpful comments on primes in arithmetic progression. We also thank Matt Papanikolas for valuable p -adic discussions. Finally, we thank the anonymous referees for insightful comments and corrections.

4. REFERENCES

- [AKS02] Agrawal, Manindra; Kayal, Neeraj; and Saxena, Nitin, "PRIMES is in P," Ann. of Math. (2) 160 (2004), no. 2, pp. 781–793.
- [AGP94] Alford, W. R.; Granville, Andrew; and Pomerance, Carl, "There are Infinitely Many Carmichael Numbers," Ann. of Math. (2) 139 (1994), no. 3, pp. 703–722.
- [AI10] Avendaño, Martín and Ibrahim, Ashraf, "Ultrametric Root Counting," submitted for publication, also available as Math ArXiv preprint 0901.3393v3.
- [AIRR10] Avendaño, Martín; Ibrahim, Ashraf; Rojas, J. Maurice; Rusek, Korben, "Succinct Certificates and Maximal Root Counts for p -adic Trinomials and Beyond," in progress.
- [BS96] Bach, Eric and Shallit, Jeff, *Algorithmic Number Theory, Vol. I: Efficient Algorithms*, MIT Press, Cambridge, MA, 1996.
- [Ber03] Bernstein, Daniel J., "Computing Logarithm Intervals with the Arithmetic-Geometric Mean Iterations," available from <http://cr.yp.to/papers.html>.
- [BRS09] Bihan, Frederic; Rojas, J. Maurice; Stella, Case E., "Faster Real Feasibility via Circuit Discriminants," proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC 2009, July 28–31, Seoul, Korea), pp. 39–46, ACM Press, 2009.
- [CG00] Cantor, David G. and Gordon, Daniel M., "Factoring polynomials over p -adic fields," Algorithmic number theory (Leiden, 2000), pp. 185–208, Lecture Notes in Comput. Sci., 1838, Springer, Berlin, 2000.

- [CDV06] Castrick, Wouter; Denef, Jan; and Vercauteren, Frederik, “Computing Zeta Functions of Nondegenerate Curves,” International Mathematics Research Papers, vol. 2006, article ID 72017, 2006.
- [Coh94] Cohen, Henri, *A course in computational algebraic number theory*, Graduate Texts in Mathematics, 138, Springer-Verlag, Berlin, 1993.
- [Coh69] Cohen, Paul J., “Decision procedures for real and p -adic fields,” *Comm. Pure Appl. Math.* 22 (1969), pp. 131–151.
- [C-T98] Colliot-Thelene, Jean-Louis, “The Hasse principle in a pencil of algebraic varieties,” *Number theory* (Tiruchirapalli, 1996), pp. 19–39, *Contemp. Math.*, 210, Amer. Math. Soc., Providence, RI, 1998.
- [DvdD88] Denef, Jan and van den Dries, Lou, “ p -adic and Real Subanalytic Sets,” *Annals of Mathematics* (2) 128 (1988), no. 1, pp. 79–138.
- [DLPvG00] *Hilbert’s Tenth Problem: Relations with Arithmetic and Algebraic Geometry*, Papers from a workshop held at Ghent University, Ghent, November 2–5, 1999. Edited by Jan Denef, Leonard Lipshitz, Thanases Pheidas and Jan Van Geel. *Contemporary Mathematics*, 270, American Mathematical Society, Providence, RI, 2000.
- [Ede87] Edelsbrunner, Herbert, *Algorithms in combinatorial geometry*, EATCS Monographs on Theoretical Computer Science, 10, Springer-Verlag, Berlin, 1987.
- [vzGKS96] von zur Gathen, Joachim; Karpinski, Marek; and Shparlinski, Igor, “Counting curves and their projections,” *Computational Complexity* 6, no. 1 (1996/1997), pp. 64–99.
- [GKZ94] Gel’fand, Israel Moseyevitch; Kapranov, Misha M.; and Zelevinsky, Andrei V.; *Discriminants, Resultants and Multidimensional Determinants*, Birkhäuser, Boston, 1994.
- [EKL06] Einsiedler, Manfred; Kapranov, Misha M.; Lind, Doug, “Non-archimedean amoebas and tropical varieties,” *J. reine und angew. Math.* 601 (2006), pp. 139–158.
- [Kho91] Khovanski, Askold, *Fewnomials*, AMS Press, Providence, Rhode Island, 1991.
- [Lau04] Lauder, Alan G. B., “Counting solutions to equations in many variables over finite fields,” *Found. Comput. Math.* 4 (2004), no. 3, pp. 221–267.
- [Len99a] Lenstra (Jr.), Hendrik W., “Finding Small Degree Factors of Lacunary Polynomials,” *Number Theory in Progress*, Vol. 1 (Zakopane-Kóscielisko, 1997), pp. 267–276, de Gruyter, Berlin, 1999.
- [Len99b] ———, “On the Factorization of Lacunary Polynomials,” *Number Theory in Progress*, Vol. 1 (Zakopane-Kóscielisko, 1997), pp. 277–291, de Gruyter, Berlin, 1999.
- [LLL82] Lenstra, Arjen K.; Lenstra (Jr.), Hendrik W.; Lovász, L., “Factoring polynomials with rational coefficients,” *Math. Ann.* 261 (1982), no. 4, pp. 515–534.
- [LP05] Lenstra (Jr.), Hendrik W., and Pomerance, Carl, “Primality Testing with Gaussian Periods,” manuscript, Dartmouth University, 2005.
- [MW99] Maller, Michael and Whitehead, Jennifer, “Efficient p -adic cell decomposition for univariate polynomials,” *J. Complexity* 15 (1999), pp. 513–525.
- [Mig82] Mignotte, Maurice, “Some Useful Bounds,” in *Computer Algebra: Symbolic and Algebraic Computation*, 2nd ed., (edited by B. Buchberger, G. E. Collins, and R. Loos, in cooperation with R. Albrecht), Springer-Verlag 1982.
- [Mor07] Morain, Francois, “Implementing the asymptotically fast version of the elliptic curve primality proving algorithm,” *Math. Comp.* 76 (2007), pp. 493–505.
- [Pap95] Papadimitriou, Christos H., *Computational Complexity*, Addison-Wesley, 1995.
- [Pla84] Plaisted, David A., “New NP-Hard and NP-Complete Polynomial and Integer Divisibility Problems,” *Theoret. Comput. Sci.* 31 (1984), no. 1–2, 125–138.
- [Poo01a] Poonen, Bjorn, “An explicit algebraic family of genus-one curves violating the Hasse principle,” 21st Journées Arithmétiques (Rome, 2001), *J. Théor. Nombres Bordeaux* 13 (2001), no. 1, pp. 263–274.
- [Poo06] ———, “Heuristics for the Brauer-Manin Obstruction for Curves,” *Experimental Mathematics*, Volume 15, Issue 4 (2006), pp. 415–420.
- [RS02] Rahman, Qazi Ibadur; and Schmeisser, Gerhard, *Analytic Theory of Polynomials*, Clarendon Press, London Mathematical Society Monographs 26, 2002.
- [Rob00] Robert, Alain M., *A course in p -adic analysis*, Graduate Texts in Mathematics, 198, Springer-Verlag, New York, 2000.
- [Roj02] Rojas, J. Maurice, “Additive Complexity and the Roots of Polynomials Over Number Fields and p -adic Fields,” *Proceedings of ANTS-V (5th Annual Algorithmic Number Theory Symposium, University of Sydney, July 7–12, 2002)*, *Lecture Notes in Computer Science* #2369, Springer-Verlag (2002), pp. 506–515.
- [Roj04] ———, “Arithmetic Multivariate Descartes’ Rule,” *American Journal of Mathematics*, vol. 126, no. 1, February 2004, pp. 1–30.
- [Roj07a] ———, “On Interpolating Between Quantum and Classical Complexity Classes,” *Proceedings of Mathematics of Quantum Computation and Quantum Technology* (November 13–16, 2005, Texas A&M University), pp. 67–88, Taylor & Francis, 2007.
- [Roj07b] ———, “Efficiently Detecting Torsion Points and Subtori,” *proceedings of MAGIC 2005 (Midwest Algebra, Geometry, and their Interactions Conference, Oct. 7–11, 2005, Notre Dame University, Indiana)*, edited by A. Corso, J. Migliore, and C. Polini, pp. 213–233, *Contemporary Mathematics*, vol. 448, AMS Press, 2007.
- [Sch80] Schwartz, Jacob T., “Fast Probabilistic Algorithms for Verification of Polynomial Identities,” *J. of the ACM* 27, 701–717, 1980.
- [Sma86] Smale, Steve, “Newton’s Method Estimates from Data at One Point,” *The Merging of Disciplines: New Directions in Pure, Applied, and Computational Mathematics* (Laramie, Wyo., 1985), pp. 185–196, Springer, New York, 1986.

Easy Composition of Symbolic Computation Software: A New Lingua Franca for Symbolic Computation

S. Linton, K. Hammond,
A. Konovalov,
University of St Andrews
{sal,kh,alexk}
@cs.st-and.ac.uk

A. D. Al Zain,
P. Trinder
Heriot-Watt University
{ceeatia, trinder}
@macs.hw.ac.uk

P. Horn
Universität Kassel
horn@math.uni-
kassel.de

D. Roozemon
Technical Universiteit
Eindhoven
d.a.roozemon@tue.nl

ABSTRACT

We present the results of the first four years of the European research project SCIEnc (www.symbolic-computation.org), which aims to provide key infrastructure for symbolic computation research. A primary outcome of the project is that we have developed a new way of combining computer algebra systems using the Symbolic Computation Software Composability Protocol (SCSCP), in which both protocol messages and data are encoded in the OpenMath format. We describe SCSCP middleware and APIs, outline some implementations for various Computer Algebra Systems (CAS), and show how SCSCP-compliant components may be combined to solve scientific problems that can not be solved within a single CAS, or may be organised into a system for distributed parallel computations.

Categories and Subject Descriptors

I.1 [Symbolic and Algebraic Manipulation]: Miscellaneous

General Terms

Design, Standardization

Keywords

OpenMath, SCSCP, interface, coordination, parallelism

1. INTRODUCTION

A key requirement in symbolic computation is to efficiently combine computer algebra systems (CAS) to solve problems that cannot be addressed by any single system. Additionally, there is often a requirement to have CAS as a back-end of mathematical databases and web or grid services, or to

combine multiple instances of the same or different CAS for parallel computations.

There are many possible combinations. Examples include: GAP and Maple in CHEVIE for handling generic character tables [21]; Maple and the PVS theorem prover to obtain more reliable results [1]; GAP and nauty in GRAPE for fast graph automorphisms [40]; and GAP as a service for the ECLiPSe constraint programming system for symmetry-breaking in search [22]. In all these cases, interfacing to a CAS with the required functionality is far less work than re-implementing the functionality in the “home” system.

Even within a single CAS, users may need to combine local and remote instances for a number of reasons, including: remote system features which are not supported in the local operating system; a need to access large (and changing) databases; remote access to the latest development version or to the configuration at the home institution; licensing restrictions permitting only online services, etc. A common quick solution is cut-and-paste from telnet sessions and web browsers. It would, however, be more efficient and more flexible to combine local and remote computations in a way such that remotely obtained results will be plugged immediately into the locally running CAS.

Moreover, individual CPUs have almost stopped increasing in power, but are becoming more numerous. A typical workstation now has 4-8 cores, and this is only a beginning. If we want to solve larger problems in future, it will be essential to exploit multiple processors in a way that gives good parallelism for minimal programmer/user effort.

CAS authors have inevitably started to face these issues, and have addressed them in various ways. For example, a CAS may write input files for another program and invoke it; the other program then will write CAS input to a file and exit; the CAS will read it and return a result. This works, but has fairly serious limitations. A better setup might allow the CAS to interact with other programs while they run and provide a separate interface to each possible external system. The SAGE system [39] is essentially built around this approach. However, achieving this is a major programming challenge, and an interface will be broken if the other system changes its I/O format, for example.

The EU Framework 6 SCIEnc project “SCIEnc – Symbolic Computation Infrastructure in Europe” is a major 5-year project that brings together CAS developers and ex-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

perts in computational algebra, OpenMath, and parallel computations. It aims to design a common standard interface that may be used for combining computer algebra systems (and any other compatible software). Our vision is an easy, robust and reliable way for users to create and consume services implemented in any compatible systems, ranging from generic services (e.g. evaluation of a string or an OpenMath object) to specialised (e.g. lookup in the database; executing certain procedure). We have developed a simple lightweight XML-based remote procedure call protocol called **SCSCP** (*Symbolic Computation Software Composability Protocol*) in which both data and instructions are represented as OpenMath objects. SCSCP is now implemented in several computer algebra systems (see Section 2.2 for an overview) and has APIs making it easy to add SCSCP interfaces to more systems. Another important outcome of the project is the development of middleware for parallel computations, **SymGrid-Par**, which is capable of orchestrating SCSCP-compliant systems into a heterogeneous system for distributed parallel computations.

We will give an overview of these tools below. First we briefly characterise the underpinning OpenMath data encoding and the SCSCP protocol (Section 2). Then we outline SCSCP interfaces in two different systems, one open source and one commercial, and provide references for existing implementations in other systems (Section 3). After that we describe several examples that demonstrate the flexibility of the SCSCP approach and some SCSCP specific features and benefits (Section 4). We introduce several SCSCP-compliant tools for parallel computations in various environments (Section 5), before concluding (Section 6).

2. A COMPUTER ALGEBRA LINGUA FRANCA

2.1 OpenMath

In order to connect different CAS it is necessary to speak a common language, i.e., to agree on a common way of marshaling mathematical semantics. Here, the obvious choice was *OpenMath* [37], a well-established standard that has been used in similar contexts. OpenMath is a very flexible language built from only twelve language elements (integers, doubles, variables, applications etc.). The entire semantics is encapsulated in *symbols* which are defined in *Content Dictionaries* (CDs) and are strictly separate from the language itself. So, one finds the “normal” addition under the name *plus* in the CD *arith1*. A large number of CDs is available at the OpenMath website [37], such as *polyd1* for the definition and manipulation of multivariate polynomials, *group4* for cosets and conjugacy classes, etc. OpenMath was designed to be efficiently used by computers, and may be represented in several different encodings. The XML representation is the most commonly used, but there exist also a binary representation and a more human readable representation called Popcorn [29]. In the current draft of MathML 3, an OpenMath dialect (called *Strict Content MathML*) is used for the semantics layer.

2.2 SCSCP

In order to actually perform communication between two systems, it is necessary to fix a low-level communication protocol. The protocol developed in SCIENCE is called *SCSCP*. SCSCP [15] is used both to link systems directly with each

other, and also as the foundation for more advanced cluster and grid infrastructures (see Section 5). The advantage of this approach is that any system that implements SCSCP can immediately connect to all other systems that already support it. This avoids the need for special cases and minimizes repeated effort. In addition, SCSCP allows remote objects to be handled by reference so that clients may work with objects of a type that do not exist in their own system at all (see the example in Section 4.2). For example, to represent the number of conjugacy classes of a group only knowledge of integers is required, not knowledge of groups. The SCSCP protocol (currently at version 1.3) is socket-based. It uses port number 26133, as assigned by the *Internet Assigned Numbers Authority* (IANA), and XML-format messages.

3. BUILDING BLOCKS FOR CAS COMPOSITION

In this section, we briefly describe the implementation of the SCSCP protocol for two systems: GAP [18] and MuPAD [34]. The main aim of this section is to show that SCSCP is a standard that may be implemented in different ways by different CAS, taking into account their own design principles.

3.1 GAP

In the GAP system, support for OpenMath and SCSCP is implemented in two GAP packages called `OpenMath` and `SCSCP`, respectively. The `OpenMath` package [11] is an OpenMath phrasebook for GAP: it converts OpenMath to GAP and vice versa, and provides a framework that users may extend with their private content dictionaries. The `SCSCP` package [30] implements SCSCP, using the GAP `OpenMath`, `IO` [35] and `GAPDoc` [32] packages. This allows GAP to run as either an SCSCP server or client. The server may be started interactively from the GAP session or as a GAP daemon. When the server accepts a connection from the client, it starts the “accept-evaluate-return” loop:

- accepts the “`procedure_call`” message and looks up the appropriate GAP function (which should be declared by the service provider as an SCSCP procedure);
- evaluates the result (or produces a side-effect);
- replies with a “`procedure_completed`” message or returns an error in a “`procedure_terminated`” message.

The SCSCP client performs the following basic actions:

- establishes connection with server;
- sends the “`procedure_call`” message to the server;
- waits for its completion or checks it later;
- fetches the result from a “`procedure_completed`” message or enters the break loop in the case of a “`procedure_terminated`” message.

We have used this basic functionality to build a set of instructions for parallel computations using the SCSCP framework. This allows the user to send several procedure calls in parallel and then collect all results, or to pick up the first available result. We have also implemented the master-worker parallel skeleton in the same way (see Section 5.2).

A demo SCSCP server is available for test purposes at `crystal.mcs.st-andrews.ac.uk`, port 26133. This runs the development version of the GAP system plus a selection of public GAP packages. Further details, downloads, and a manual with examples are available online [30].

3.2 MuPAD

There are two main aspects to the MuPAD SCSCP support: the MuPAD `OpenMath` package [26] and the SCSCP server wrapper for MuPAD. The former offers the ability to parse, generate, and handle OpenMath in MuPAD and to consume SCSCP services, the latter provides access to MuPAD's mathematical abilities as an SCSCP service. The current MuPAD end-user license agreement, however, does not generally allow providing MuPAD computational facilities over the network. We therefore focus on the open-source `OpenMath` package, which can be downloaded from [26].

3.2.1 OpenMath Parsing and Generation

Two functions are available to convert an OpenMath XML string into a tree of MuPAD `OpenMath::` objects, namely `OpenMath::parse(str)` which parses the string `str`, and `OpenMath::parseFile(fname)` which reads and parses the file named `fname`. Conversely, a MuPAD expression can be converted into its OpenMath representation using `generate::OpenMath`. Note that it is not necessary to directly use OpenMath in MuPAD if the SCSCP connection described below is used: the package takes care of marshalling and unmarshalling in a way that is completely transparent to the MuPAD user.

3.2.2 SCSCP Client Connection

The call `s := SCSCP(host, port)` creates an SCSCP connection object, that can subsequently be used to send commands to the SCSCP server. Note that the actual connection is initiated on construction by starting the Java program WUPSI [27] which is bundled with the `OpenMath` package. This uses an asynchronous message-exchange mode, and can therefore be used to introduce background computations. The command `s::compute(...)` can then be used to actually compute something on the server (`s(...)` is equivalent). Note that it may be necessary to wrap the parameter in `hold(...)` to prevent premature evaluation on the client side. In order to use the connection asynchronously, the `send` and `retrieve` commands may be used: `a := s::send(...)` returns an integer which may be used to identify the computation. The result may subsequently be retrieved using `s::retrieve(a)`. `retrieve` will normally return `FAIL` if the result of the computation is not yet computed, but this behaviour can be overridden using a second parameter to force the call to block.

3.3 Other Implementations of SCSCP

The SCIENCE project has produced a Java library [28] that acts as a reference implementation for systems developers who would like to implement SCSCP for their own systems. This is freely available under the Apache2 license. In addition to GAP and MuPAD, SCSCP has also been implemented in two other systems participating in the SCIENCE project: KANT [17] and Maple [33] (the latter implementation is currently a research prototype and not available in the Maple release). There are third-party implementations for TRIP [19, 20], Magma [8] (as a wrapper application), and

Macaulay2 [24]. SCSCP thus, as intended, allows a large range of CAS to interact and to share computations.

4. EXAMPLES

In this section we provide a number of examples which demonstrate the features and benefits of SCSCP, such as flexible design, composition of different CAS, working with remote objects and speeding up computations. More examples can be found in e.g. [14, 16] and on the web sites for individual systems.

4.1 GAP

In order to illustrate the flexibility of our approach, we will describe three possible ways to set up a procedure for the same kind of problems.

The GAP Small Groups Library [7] contains all groups of orders up to 2000, except groups of order 1024. The GAP command `SmallGroup(n,i)` returns the i -th group of order n . Moreover, for any group G of order $1 \leq |G| \leq 2000$ where $|G| \notin \{512, 1024\}$, GAP can determine its *library number*: the pair $[n,i]$ such that G is isomorphic to `SmallGroup(n,i)`. This is in particular the most efficient way to check whether two groups of “small” order are isomorphic or not.

Let us consider now how we can provide a group identification service with SCSCP. When designing an SCSCP procedure to identify small groups, we first need to decide how the client should transmit a group to the server. We will give three possible scenarios and outline simple steps needed for the design and provision of the SCSCP services within the provided framework.

Case 1. A client supports permutation groups (for example, a client is a minimalistic GAP installation without the Small Groups Library). In this case the conversion of the group to and from OpenMath will be performed straightforwardly, so that the service provider only needs to install the function `IdGroup` as an SCSCP procedure (under the same or different name) before starting the server:

```
gap> InstallSCSCPprocedure("IdGroup", IdGroup);
InstallSCSCPprocedure : IdGroup installed.
```

The client may then call this, obtaining a record with the result in its object component:

```
gap> EvaluateBySCSCP("IdGroup", [SymmetricGroup(6)],
> "scscp.st-and.ac.uk", 26133);
rec( attributes := [ [ "call_id", "hp0SE18S" ] ],
      object := [ 720, 763 ] )
```

Case 2. A client supports matrices, but not matrix groups. In this case, the service provider may install the SCSCP procedure which constructs a group generated by its arguments and return its library number:

```
IdGroupByGens := gens -> IdGroup( Group( gens ) );
```

Note that validity of any input and the applicability of the `IdGroup` method to the constructed group will be automatically checked by GAP during the execution of the procedure on the SCSCP server, so there is no need to add such checks to this procedure (though they may be added to replace the standard GAP error message for these cases by other text).

Case 3. A client supports groups in some specialised representation (for example, groups given by pc-presentation in GAP). Indeed, for groups of order 512 the Small Groups Library contains all 10494213 non-isomorphic groups of this

order and allows the user to retrieve any group by its library number, but it does not provide an identification facility. However, the GAP package ANUPQ [36] provides a function `IdStandardPresented512Group` that performs the latter task. Because the ANUPQ package only works in a UNIX environment it is useful to design an SCSCP service for identification of groups of order 512 that can be called from within GAP sessions running on other platforms (note that the client version of the SCSCP package for GAP does work under Windows).

Now the problem reduces to the encoding of such a group in OpenMath. Should it, for example, be converted into a permutation representation, which can be encoded using existing content dictionaries or should we develop a new content dictionary for groups in such a representation? Luckily, the SCSCP protocol provides enough freedom for the user to select his/her own data representation. Since we are interfacing between two copies of the GAP system, we are free to use a GAP-specific data format, namely the *pcgs code*, an integer that describes the *polycyclic generating sequence* (*pcgs*) of the group, to pass the data to the server (see the GAP manual and [6] for more details).

First we create a function that takes the *pcgs code* of a group of order 512 and returns the number of this group in the GAP Small Groups library:

```
gap> IdGroup512 := function( code )
>   local G, F, H;
>   G := PcGroupCode( code, 512 );
>   F := PqStandardPresentation( G );
>   H := PcGroupFpGroup( F );
>   return IdStandardPresented512Group( H );
> end;;
```

After such a function is created on the server, it becomes “visible” as an SCSCP procedure under the same name:

```
gap> InstallSCSCPprocedure("IdGroup512",IdGroup512);
InstallSCSCPprocedure : IdGroup512 installed.
```

For convenience, the client may be supplied with a function that is specialised to use the correct server port, and which checks that the transmitted group is indeed of order 512:

```
gap> IdGroup512Remote:=function( G )
>   local code, result;
>   if Size(G)<>512 then Error("|G|<>512\n");fi;
>   code := CodePcGroup( G );
>   result := EvaluateBySCSCP("IdGroup512",[code],
>   "scscp.st-and.ac.uk", 26133);
>   return result.object;
> end;;
```

Now the call to `IdGroup512Remote` returns the result in the standard `IdGroup` notation:

```
gap> IdGroup512Remote( DihedralGroup( 512 ) );
[ 512, 2042 ]
```

4.2 GAP and Macaulay2

We now consider interaction between GAP and Macaulay2 [24]. Macaulay2 is “a software system devoted to supporting research in algebraic geometry and commutative algebra,” that is particularly well known for its efficient Gröbner bases procedures. We have implemented OpenMath and the SCSCP protocol as packages in the Macaulay2 system, and they have been available in the stable branch since late 2009. The OpenMath support includes basic arithmetic, matrices, finite fields elements, polynomials, Gröbner bases, etc. All

support for OpenMath symbols was implemented directly in the Macaulay2 language, to allow for easy maintenance and extensibility.

Macaulay2 is fully SCSCP 1.3 compatible and can act both as a server and as a client. The server is multithreaded so it can serve many clients at the same time, and supports storing and retrieving of remote objects. The client was designed in such a way as to disclose remote computation using SCSCP with minimal interaction from the user. It supports convenient creation and handling of remote objects, as demonstrated below.

An example of a GAP client calling a Macaulay2 server for the Gröbner basis computation, can be found [16]. Although this 2008 implementation used a prototype wrapper implementation of an SCSCP server for Macaulay2, rather than the full internal implementation that we have now, it nicely demonstrates the possible gain of connecting computer algebra systems using SCSCP.

The next example of a Macaulay2 SCSCP client calling a remote GAP server was produced using the current implementation. First, we load the OpenMath and SCSCP packages and establish a connection to the GAP server that accepts and evaluates OpenMath objects.

```
i1 : loadPackage "SCSCP"; loadPackage "OpenMath";

i3 : GAP = newConnection "127.0.0.1"
o3 = SCSCP Connection to GAP (4.dev) on
      scscp.st-and.ac.uk:26133

o3 : SCSCPConnection
```

We demonstrate the conversion of an arithmetic operation to OpenMath syntax (note the abbreviated form Macaulay2 uses to improve legibility of XML expressions), and evaluate the expression in GAP.

```
i4 : openMath 1+2
o4 = <OMA
      <OMS cd="arith1" name="plus"
      <OMI "1"
      <OMI "2"
o4 : XMLnode
```

```
i5 : GAP <== openMath 1 + 2
o5 = 3
```

We then create two matrices in Macaulay2 (suppressing the output of the creation of the second one), and create a remote object *G* in GAP representing the group they generate.

```
i6 : m1 = id_(QQ^10)~{1,6,2,7,3,8,4,9,5,0}
o6 = | 0 1 0 0 0 0 0 0 0 0 |
      | 0 0 0 0 0 0 1 0 0 0 |
      | 0 0 1 0 0 0 0 0 0 0 |
      | 0 0 0 0 0 0 0 1 0 0 |
      | 0 0 0 1 0 0 0 0 0 0 |
      | 0 0 0 0 0 0 0 0 1 0 |
      | 0 0 0 0 1 0 0 0 0 0 |
      | 0 0 0 0 0 0 0 0 0 1 |
      | 0 0 0 0 0 1 0 0 0 0 |
      | 1 0 0 0 0 0 0 0 0 0 |
      10      10
o6 : Matrix QQ <--- QQ
```

```
i7 : m2 = id_(QQ^10)~{1,0,2,3,4,5,6,7,8,9};
```

```
i8 : G = GAP <=== matrixGroup({m1,m2})
```

```
o8 = << Remote GAP object >>
o8 : RemoteObject
```

When we ask for the size of the group, Macaulay2 simply creates a new object representing $|G|$. Finally, evaluating this object in GAP gives the number of elements in the group generated by those matrices.

```
i9 : size G
o9 = << Remote GAP object >>
o9 : RemoteObject
```

```
i10 : GAP <== size G
o10 = 10080
```

One of the most important features of this example is that despite the fact that Macaulay2 has no support for groups at all, by using OpenMath and SCSCP, we can still create an object that represents a group, and obtain useful information about it.

4.3 MuPAD

To show how the OpenMath MuPAD package is used, we first demonstrate some features of the OpenMath package:

```
>> package("OpenMath"):
>> 1+a*sin(x)
a*sin(x) + 1
>> om := OpenMath(%)
arith1.plus(1, arith1.times(transc1.sin($x), $a))
>> OpenMath::toXml(om)
<OMA>
  <OMS cd='arith1' name='plus' />
  <OMI>1</OMI>
</OMA>
  <OMS cd='arith1' name='times' />
  <OMA>
    <OMS cd='transc1' name='sin' />
    <OMV name='x' />
  </OMA>
  <OMV name='a' />
</OMA>
```

Now we use it to establish an SCSCP connection to a machine 400km away that is running KANT [12]. We use the KANT server to factor the product of shifted Swinnerton-Dyer polynomials. Of course, we could do it locally in MuPAD, but that would take 38 seconds:

```
>> swindyer := proc(plist) ... :
>> R := Dom::UnivariatePolynomial(x, Dom::Rational):
>> p1 := R(swindyer([2,3,5,7,11])):
>> p2 := R(subs(swindyer([2,3,5,7,13,17])), x=3*x-2):
>> p := p1 * p2:
>> degree(p), nterms(p)
96, 49
>> st := time(): F1 := factor(p): time()-st
38431
```

Now let us use KANT remotely:

```
>> package("OpenMath"):
>> kant := SCSCP("scscp.math.tu-berlin.de", 26133):
>> st:=rtime():
  F2:=kant::compute(hold(factor)(p)):
  rtime()-st
1221
```

Establishing the connection, marshalling and unmarshalling the objects, sending them over the network, and the actual KANT computation took only 1.2 seconds in total. This demonstrates the flexibility of the SCSCP approach: the most appropriate system may be used for the task at hand. Users are no longer restricted to performing all aspects of a required computation in a single system that may not provide good support for all required operations.

5. INFRASTRUCTURE FOR PARALLEL COMPUTATIONS

In contrast to notations for numerical computations, which have an emphasis on floating point arithmetic, monolithic arrays, and programmer-controlled memory allocation, symbolic computing has an emphasis on functional notations, greater interactivity, very high level programming abstractions, complex data structures, automatic memory management, etc. With this different evolutionary path, it is not surprising that symbolic computation has parallelisation requirements that differ significantly from those for traditional *numerical* high-performance computing. In particular, parallel symbolic computations are often highly irregular, need to exploit more complex data structures than their numerical counterparts, and exhibit sophisticated computational patterns that are only just being identified.

We have developed a number of tools that exploit the capabilities of SCSCP for marshaling/unmarshaling symbolic data as part of a parallel computation, outlined below: SPSPD (a middleware written in Java using the SCSCP API); the master-worker skeleton implemented directly in GAP; and a general programmable framework for parallelism, **SymGrid-Par**. These provide increasing levels of capability and scalability.

5.1 WUPSI/SPSD

The Java framework outlined above [28] has been used to construct “WUPSI”, an integrating software component that is a universal Popcorn SCSCP Interface providing several different technologies for interacting with SCSCP clients and servers. One of these is the Simple Parallel SCSCP Dispatcher (SPSD), which allows very simple patterns like parallel map or zip to be used on different SCSCP servers simultaneously. The parallelization functionality is offered as an SCSCP service itself, so it can be invoked not only from the WUPSI command line, but also by any other SCSCP client. Since WUPSI and all parts of it are open source and freely available, they can be exploited to build whatever infrastructure seems necessary for a specific use case.

5.2 GAP Master-Worker Skeleton

Using the SCSCP package for GAP, it is possible to send requests to multiple services to execute them in parallel, or to wait until the fastest result is available, and implement various scenarios on top of the provided functionality. One of these is the master-worker skeleton, included in the package and implemented purely in GAP. The client (i.e. master, which orchestrates the computation) works in any system that is able to run GAP, and it may even orchestrate both GAP based and non-GAP based SCSCP servers, exploiting such SCSCP mechanisms as transient content dictionaries to define OpenMath symbols for a particular operation that exists on a specific SCSCP server, and remote objects to keep references to objects that may be supported only on the

other CAS. It is quite robust, especially for stateless services: if a server (i.e. worker) is lost, it will resubmit the request to another available server. Furthermore, it allows new workers (from a previously declared pool of potential workers) to be added during the computation. It has flexible configuration options and produces parallel trace files that can be visualised using EdenTV [5]. The master-worker skeleton shows almost linear (e.g. 7.5 on 8-core machine) speedup on irregular applications with low task granularity and no nested parallelism. The SCSCP package manual [30] contains further details and examples. See also [13, 31] for two examples of using the package to deal with concrete research problems.

5.3 SymGrid-Par

SymGrid [25] provides a new framework for executing symbolic computations on *computational Grids*: distributed parallel systems built from geographically-dispersed parallel clusters of possibly heterogeneous machines. It builds on and extends standard Globus toolkit [23] capabilities, offering support for discovering and accessing Web and Grid-based symbolic computing services (**SymGrid-Services** [9]) and for orchestrating symbolic components into Grid-enabled applications (**SymGrid-Par** [2]). Both of these components build on SCSCP in an essential way. Below, we will focus on **SymGrid-Par**, which aims to orchestrate multiple sequential symbolic computing engines into a single coherent parallel system.

5.3.1 Implementation details

SymGrid-Par (Figure 1) extends our implementation of the GUM system [4, 41], a message-based portable parallel implementation of the widely used purely functional language Haskell [38] for both shared and distributed memory architectures. **SymGrid-Par** comprises two generic interfaces: the “*Computational Algebra system to Grid middleware*” (**CAG**) interface links a CAS to GUM; and the “*Grid middleware to Computational Algebra system*” (**GCA**) interface conversely links GUM to a CAS. The CAG interface is used by computational algebra systems to interact with GUM. GUM then uses the GCA interface to invoke remote computational algebra system functions, to communicate with the CAS etc. In this way, we achieve a clear separation of concerns: GUM deals with issues of thread creation/coordination and orchestrates the CAS engines to work on the application as a whole; while each instance of the CAS engine deals solely with execution of individual algebraic computations.

The GCA interface interfaces our middleware with a CAS, connecting to a small interpreter that allows the invocation of arbitrary computational algebra system functions, marshaling/unmarshaling data as required. The interface comprises both C and Haskell components. The C component is mainly used to invoke operating system services that are needed to initiate the computational algebra process, to establish communication channels, and to send and receive commands/results from the computational algebra system process. It also provides support for static memory that can be used to maintain state between calls. The Haskell component provides interface functions to the user program and implements the communication protocol with the computational algebra process.

The CAG interface comprises an API for each symbolic system that provides access to a set of common (and po-

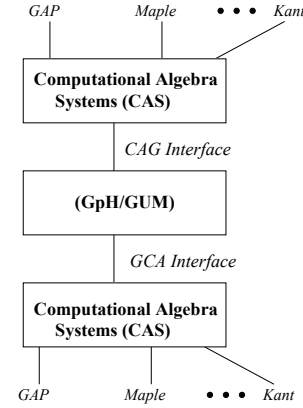


Figure 1: SymGrid-Par Design Overview

tentially parallel) patterns of symbolic computation. These patterns form a set of dynamic *algorithmic skeletons* (see [10]), which may be called directly from within the computational algebra system, and which may be used to orchestrate a set of sequential components into a parallel computation. In general (and unlike most skeleton approaches), these patterns will be nested and can be dynamically composed to form the required parallel computation. Also, in general, they may mix components taken from several different computational algebra systems.

5.3.2 Standard Parallel Patterns

The standard patterns we have identified are listed below. The patterns are based on commonly-used sequential higher-order functions that can be found in functional languages such as Haskell. Similar patterns are often defined as algorithmic skeletons. Here, each argument to the pattern is separated by an arrow ($\rightarrow$), and may operate over lists of values ($[..]$), or pairs of values ($(.., ..)$). All of the patterns are *polymorphic*: i.e. a, b etc. stand for (possibly different) concrete types. The first argument in each case is a function of either one or two arguments that is to be applied in parallel.

```

parMap::      (a->b) -> [a] -> [b]
parZipWith::  (a->b->c) -> [a] -> [b] -> [c]
parReduce::   (a->b->b) -> b -> [a] -> b
parMapReduce::(d->[a]->b) -> (c->[(d,a)]) -> c -> [(d,b)]
masterSlaves::((a->a)->(a->a->b)) -> [(a,a)] -> [(a,a,b)]

```

So, for example, **parMap** is a pattern taking two arguments and returning one result. Its first argument (of type $a \rightarrow b$) is a function from some type a to some other type b , and its second argument (of type $[a]$) is a list of values of type a . It returns a list of values each of type b . Operationally, **parMap** applies a function argument to each element of a list, in parallel, returning the list of results, e.g.

```

parMap double [1,4,9,16] == [2,8,18,32]
where double x = x + x

```

It thus implements a parallel version of the common **map** function, which applies a function to each element of list. The **parZipWith** pattern similarly applies a function, but in this case to two arguments, one taken from each of its list arguments. Each application is performed in parallel, e.g.

```

parZipWith add [1,4,9,16] [3,5,7,9] == [4,9,16,25]
where add x y = x + y

```

Again, this implements a parallel version of the `zipWith` function that is found in functional languages such as Haskell. Finally, `parReduce` reduces its third argument (a list of type `[a]`) by applying a function (of type `a->b->b`) between pairs of its elements, ending with the value of the same type `b` as its second argument; `parMapReduce` pattern combines features of both `parMap` and `parReduce`, first generating a list of key-value pairs from every input item (in parallel), before reducing each set of values for one key across these intermediate results; `masterSlaves` is used to introduce a set of tasks and generate a set of worker processes to apply the given function parameter in parallel to these tasks under the control of a coordinating master task. The `parReduce` and `parMapReduce` patterns are often used to construct parallel pipelines, where the elements of the list will themselves be lists, perhaps constructed using other parallel patterns. In this way, we can achieve nested parallelism. [3] contains further details on **SymGrid-Par**, including the description of several experiments and a detailed analysis of their parallel performance.

6. CONCLUSIONS

We have presented a framework for combining computer algebra systems using a newly-developed remote procedure call protocol SCSCP (Symbolic Computation Software Composability Protocol). By defining common data and task interfaces for all systems, we allow complex computations to be constructed by orchestrating heterogeneous distributed components into a single symbolic application. Any system supporting SCSCP can immediately connect to all other SCSCP-compliant systems, thus avoiding the need for special cases and minimizing repeated efforts. Furthermore, if some CAS changes its internal format then it only needs to update one interface, namely that to the SCSCP protocol (instead of as many interfaces as there are programs it connects to). Moreover, this change can take place completely transparently to the other CAS connecting to it.

We have demonstrated several examples of setting up communication between different CAS, thus exhibiting SCSCP benefits and features including its flexible design, the ability to solve problems that can not be solved in the “home” system, and the possibility to speed up computations by sending request to a faster CAS. Finally, we have shown how sequential systems can be combined into heterogeneous parallel systems that can deliver good parallel performance.

SCSCP uses an OpenMath representation to encode both transmitted data and protocol instructions, and may be supported not only by a CAS, but by any other software as well. To achieve this, it is necessary only to support SCSCP messages accordingly to the protocol specification, while the support of particular OpenMath constructions and objects is dictated only by the nature of the application. This support may consequently be limited to a few basic OpenMath data types and a small set of application-relevant symbols. For example, a Java applet to display the lattice of subgroups of a group may be able to draw diagrams for partially ordered sets without any support for the group-theoretical OpenMath CDs. Other possible applications may include a web or SCSCP interface to a mathematical database, or, as an extreme proof-of-concept, even a server providing access to a computer algebra system through an Internet Relay Chat bot.

Additionally, SCSCP-compliant middleware may look in-

side an SCSCP message, extracting all necessary technical information from its outer levels and taking the embedded OpenMath objects as a “black box”. This approach is essentially used in the **SymGrid-Par** middleware (Section 5) which performs marshaling and unmarshaling of OpenMath-represented data between CASes. By exploiting well-established adaptive middleware (GUM), we can manage complex irregular parallel computations on clusters and shared-memory parallel machines. This allows us to harness a number of advanced GUM features that are important to symbolic computations, including: automatic control of task granularity, dynamic task creation, implicit asynchronous communication, automatic sharing-preserving data marshaling/unmarshaling, ultra-lightweight work stealing and task migration, virtual shared memory, and distributed garbage collection.

We have already seen examples of SCSCP-compliant software that were created outside the SCIENCE project and we hope that we will have more of them in the future. We anticipate that existing and emerging SCSCP APIs will be useful here as templates for new APIs. In conclusion, SCSCP is a powerful and flexible framework for combining CAS, and we encourage developers to cooperate with us in adding SCSCP support to their software.

7. ACKNOWLEDGMENTS

“SCIENCE – Symbolic Computation Infrastructure in Europe” (www.symbolic-computation.org) is supported by EU FP6 grant RII3-CT-2005-026133. We would like to acknowledge the fruitful collaborative work of all partners and CAS developers involved in the project.

8. REFERENCES

- [1] A. Adams, M. Dunstan, H. Gottlieb, T. Kelsey, U. Martin, and S. Owre. Computer Algebra meets Automated Theorem Proving: Integrating Maple and PVS. In *Proc. TPHOLs 2001: Intl. Conf. on Theorem Proving in Higher Order Logics, Springer LNCS 2152*, pages 27–42, 2001.
- [2] A. Al Zain, K. Hammond, P. Trinder, S. Linton, H.-W. Loidl, and M. Costantini. SymGrid-Par: Designing a Framework for Executing Computational Algebra Systems on Computational Grids. In *Proc. ICCS '07: 7th Intl. Conf. on Computational Science, Springer LNCS 4488*, pages 617–624, 2007.
- [3] A. Al Zain, P. Trinder, K. Hammond, A. Konovalov, S. Linton, and J. Berthold. Parallelism without pain: Orchestrating computational algebra components into a high-performance parallel system. In *Proc. IEEE Intl. Symp. on Parallel and Distributed Processing with Applications (ISPA 2008), Sydney, Australia*, pages 99–112, 2008.
- [4] A. Al Zain, P. Trinder, G. Michaelson, and H.-W. Loidl. Evaluating a High-Level Parallel Language (GpH) for Computational GRIDS. *IEEE Trans. Parallel Distrib. Syst.*, 19(2):219–233, 2008.
- [5] J. Berthold and R. Loogen. Visualizing parallel functional program runs: Case studies with the eden trace viewer. In *Proc. PARCO 2007: Intl. Conf. on Parallel Computing: Architectures, Algorithms and Applications*, volume 15 of *Advances in Parallel Computing*, pages 121–128. IOS Press, 2007.

- [6] H. Besche and B. Eick. Construction of finite groups. *J. Symbolic Comput.*, 27(4):387–404, 1999.
- [7] H. Besche, B. Eick, and E. O’Brien. *The Small Groups Library*. <http://www-public.tu-bs.de:8080/~beick/soft/small/small.html>.
- [8] J. Cannon and W. Bosma (Eds.). *Handbook of Magma Functions, Edition 2.15*, 2008. School of Mathematics and Statistics, University of Sydney <http://magma.maths.usyd.edu.au/>.
- [9] A. Cârstea, M. Frîncu, G. Macariu, D. Petcu, and K. Hammond. Generic Access to Web and Grid-based Symbolic Computing Services: the SymGrid-Services Framework. In *Proc. ISPDC 07: Intl. Symp. on Parallel and Distributed Computing, Castle Hagenberg, Austria, IEEE Press*, pages 143–150, 2007.
- [10] M. Cole. Algorithmic Skeletons. In *Research Directions in Parallel Functional Programming*, chapter 13, pages 289–304. Springer-Verlag, 1999.
- [11] M. Costantini, A. Konovalov, and A. Solomon. *OpenMath – OpenMath functionality in GAP, Version 10.1*, 2010. GAP package. <http://www.cs.st-andrews.ac.uk/~alexk/openmath.htm>.
- [12] M. Daberkow, C. Fieker, J. Klüners, M. Pohst, K. Roegner, M. Schörmig, and K. Wildanger. KANT V4. *J. Symbolic Comput.*, 24(3-4):267–283, 1997. Computational Algebra and Number Theory, 1993.
- [13] B. Eick and A. Konovalov. The modular isomorphism problem for the groups of order 512. In *Groups St. Andrews 2009*, London Math. Soc. Lecture Note Ser. (Accepted).
- [14] S. Freundt, P. Horn, A. Konovalov, S. Lesseni, S. Linton, and D. Roozmond. OpenMath in SCIENCE: Evolving of symbolic computation interaction. In proceedings of OpenMath Workshop 2009 (to appear).
- [15] S. Freundt, P. Horn, A. Konovalov, S. Linton, and D. Roozmond. Symbolic Computation Software Composability Protocol (SCSCP) specification. <http://www.symbolic-computation.org/scscp>, Version 1.3, 2009.
- [16] S. Freundt, P. Horn, A. Konovalov, S. Linton, and D. Roozmond. Symbolic computation software composability. In *AISC/MKM/Calculemus, Springer LNCS 5144*, pages 285–295, 2008.
- [17] S. Freundt and S. Lesseni. *KANT 4 SCSCP Package*. <http://www.math.tu-berlin.de/~kant/kantscscp.html>.
- [18] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4.12*, 2008. <http://www.gap-system.org>.
- [19] M. Gastineau. *SCSCP C Library – A C/C++ library for Symbolic Computation Software Composability Protocol, Version 0.6.0*. IMCCE, 2009. <http://www.imcce.fr/Equipes/ASD/trip/scscp/>.
- [20] M. Gastineau. Interaction between the specialized and general computer algebra systems using the SCSCP protocol. Submitted.
- [21] M. Geck, G. Hiss, F. Lübeck, G. Malle, and G. Pfeiffer. CHEVIE – A system for computing and processing generic character tables for finite groups of Lie type, Weyl groups and Hecke algebras. *Appl. Algebra Engrg. Comm. Comput.*, 7:175–210, 1996.
- [22] I. Gent, W. Harvey, T. Kelsey, and S. Linton. Generic SBDD Using Computational Group Theory. In *Proc. CP 2003: Intl. Conf. on Principles and Practice of Constraint Programming, Kinsale, Ireland*, pages 333–347, 2003.
- [23] Globus toolkit. <http://www.globus.org/toolkit/>.
- [24] D. Grayson and M. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at <http://www.math.uiuc.edu/Macaulay2/>.
- [25] K. Hammond, A. Al Zain, G. Cooperman, D. Petcu, and P. Trinder. SymGrid: a Framework for Symbolic Computation on the Grid. In *Proc. EuroPar’07*, LNCS, Rennes, France, August 2007.
- [26] P. Horn. *MuPAD OpenMath Package*, 2009. <http://mupad.symcomp.org/>.
- [27] P. Horn and D. Roozmond. *java.symcomp.org – Java Library for SCSCP and OpenMath*. <http://java.symcomp.org/>.
- [28] P. Horn and D. Roozmond. *WUPSI – Universal Popcorn SCSCP Interface*. <http://java.symcomp.org/wupsi.html>.
- [29] P. Horn and D. Roozmond. OpenMath in SCIENCE: SCSCP and POPCORN. In *Intelligent Computer Mathematics – MKM 2009*, volume 5625 of *Lecture Notes in Artificial intelligence*, pages 474–479. Springer, 2009.
- [30] A. Konovalov and S. Linton. *SCSCP – Symbolic Computation Software Composability Protocol, Version 1.2*, 2010. GAP package. <http://www.cs.st-andrews.ac.uk/~alexk/scscp.htm>.
- [31] A. Konovalov and S. Linton. Parallel computations in modular group algebras. (Submitted, 2010).
- [32] F. Lübeck and M. Neunhöffer. *GAPDoc – A Meta Package for GAP Documentation*, 2008. <http://www.math.rwth-aachen.de/~Frank.Luebeck/GAPDoc>.
- [33] Maple. <http://www.maplesoft.com/>.
- [34] MuPAD. <http://www.sciface.com/>.
- [35] M. Neunhöffer. *IO – Bindings for low level C library IO*, 2009. <http://www-groups.mcs.st-and.ac.uk/~neunhoefer/Computer/Software/Gap/io.html>.
- [36] E. O’Brien, W. Nickel, and G. Gamble. *ANUPQ – ANU p-Quotient, Version 3.0*, 2006. <http://www.math.rwth-aachen.de/~Greg.Gamble/ANUPQ/>.
- [37] OpenMath. <http://www.openmath.org/>.
- [38] S. Peyton Jones (ed.), J. Hughes, L. Augustsson, D. Barton, B. Boutel, W. Burton, J. Fasel, K. Hammond, R. Hinze, P. Hudak, T. Johnsson, M. Jones, J. Launchbury, E. Meijer, J. Peterson, A. Reid, C. Runciman, and P. Wadler. *Haskell 98 Language and Libraries. The Revised Report*. Cambridge University Press, April 2003.
- [39] Sage. <http://www.sagemath.org/>.
- [40] L. Soicher. Computing with graphs and groups. In *Topics in Algebraic Graph Theory*, pages 250–266. Cambridge University Press, 2004.
- [41] P. Trinder, K. Hammond, J. Mattson Jr., A. Partridge, and S. Peyton Jones. GUM: a Portable Parallel Implementation of Haskell. In *Proc. PLDI ’96: Intl. Conf. on Programming Language Design and Implementation, Philadelphia, PA, USA*, pages 79–88, May 1996.

Symbolic Integration at Compile Time in Finite Element Methods

Karl Rupp

Christian Doppler Laboratory for Reliability Issues in Microelectronics
at the Institute for Microelectronics, TU Wien
Gußhausstraße 27–29/E360
A-1040 Wien, Austria
rupp@iue.tuwien.ac.at

ABSTRACT

In most existing software packages for the finite element method it is not possible to supply the weak formulation of the problem of interest in a compact form, which was in the early days of programming due to the low abstraction capabilities of available programming languages. With the advent of pure object-oriented programming, abstraction was long said to be achievable only in trade-off with run time efficiency. In this work we show that it is possible to obtain both a high level of abstraction and good run time efficiency by the use of template metaprogramming in C++. We focus on a mathematical expressions engine, by which element matrices are computed during compile time and by which the weak formulation can be specified in a single line of code. A comparison of system matrix assembly times of existing finite element software shows that the template metaprogramming approach is up to an order of magnitude faster than traditional software designs.

Categories and Subject Descriptors

D.1.m [Programming Techniques]: Miscellaneous; G.1.4 [Numerical Analysis]: Quadrature and Numerical Differentiation—*Automatic differentiation*; G.1.8 [Numerical Analysis]: Partial Differential Equations—*Finite element methods*; G.4 [Mathematical Software]: Efficiency; I.1.3 [Symbolic and Algebraic Manipulation]: Languages and Systems—*Special-purpose algebraic systems*

General Terms

Design, Languages

Keywords

Template Metaprogramming, Symbolic Integration, Finite Element Methods, C++

1. INTRODUCTION

The level of abstraction in most software packages dealing with the finite element method (FEM) is low, mainly because for a long time programming languages could not provide facilities for a higher level of abstraction and thus low level programming approaches are extensively documented in the literature. With the advent of pure object-oriented programming, abstraction was long said to be achievable only in trade-off with run time efficiency, which is again one of the major aims of scientific software. In order to still achieve a reasonably high level of abstraction and good run time efficiency, program generators have been designed to parse higher level descriptions and to generate the required source code, which is then compiled into an executable form. Examples for such an approach are freeFEM++ [6] or DOLPHIN [5].

The introduction of another layer in the compilation process is in fact not very satisfactory: On the one hand, an input file syntax needs to be specified and parsed correctly, and on the other hand, proper source-code should be generated for all semantically valid inputs. Moreover, it gets much harder to access or manipulate objects at source code level, because any modifications in the higher level input file causes another precompilation run, potentially producing entirely different source code. Thus, it pays off to avoid any additional external precompilation, and instead provide higher-level components directly at source code level. To the author's knowledge, the highest level of abstraction for FEM at source code level has so far been achieved by Sundance [10], which heavily relies on object oriented programming to raise the level of abstraction directly at source code level while reducing run time penalties to a minimum.

In this work we present a compile time engine for mathematical expressions obtained through template metaprogramming [1], so that the level of abstraction at source code level effectively meets that of the underlying mathematical description. Additionally, due to dispatches at compile time, any penalties due to virtual dispatches at run time are avoided. Since both the weak formulation of the underlying mathematical problem and the test and trial functions are available at compile time, we evaluate local element matrices symbolically by the compiler, so any unnecessary numerical integration at run time is avoided. Our approach only relies on facilities provided with standard-conforming C++ compilers to generate the appropriate code that finally enters the executable, thus no further external dependencies have to be fulfilled.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

As driving example throughout this work we consider the Poisson equation

$$-\Delta u = 1 \quad (1)$$

in a domain Ω with, say, homogeneous Dirichlet boundary conditions. However, the techniques presented in the following can also be applied to more general problems with additional flux terms, a more complicated right hand side or different types of boundary conditions. The weak formulation of (1) is to find u in a suitable trial space such that

$$a(u, v) := \int_{\Omega} \nabla u \nabla v \, dx = \int_{\Omega} v \, dx =: L(v) \quad (2)$$

for all test functions v in a certain test space. After discretization, the resulting system matrix $\mathbf{S}$ is given by

$$\mathbf{S} = (S_{i,j})_{i,j=1}^N, \quad S_{i,j} = a(\varphi_j, \psi_i), \quad (3)$$

where φ_j and ψ_i are the trial and test functions from the trial and test spaces respectively [3,13]. $\mathbf{S}$ is typically sparse due to the local support of the chosen basis functions.

In the following we assume that the system matrix $\mathbf{S}$ is fully set up prior to solving the resulting system of linear equations. Since typically iterative solvers are used for the solution of the linear system, it is in principle sufficient to provide matrix-vector multiplications and never set up the full system matrix. Our approach is also suitable for such a configuration, but for better comparison with other software packages in Sec. 5 and Sec. 6 we consider the case that $\mathbf{S}$ is set up explicitly.

According to (2) and (3), a generic finite element implementation must be able to evaluate bilinear forms for varying function arguments. Moreover, since the bilinear form is to be supplied by the user, its specification should be as easy and as convenient as possible. Consequently, we start with the discussion of higher-level components for the specification of the weak formulation in Sec. 2. The specification and manipulation of test and trial functions is outlined in Sec. 3. In Sec. 4 the higher-level components are joined in order to compute local element matrices at compile time. The influence on compilation times and execution times is quantified in Sec. 5 and Sec. 6 respectively.

2. EXPRESSION ENGINE

We have implemented high level components for the compile time representation of mathematical expressions in the style of *expression templates* [11,12]. The concept of syntax trees often used at run time was adapted to handle operators and operands at compile time:

```
1 template <typename ScalarType,
2         typename LHS,
3         typename RHS,
4         typename OP >
5 class Expression;
```

`ScalarType` denotes the underlying integral type used for the arithmetic operations, `LHS` and `RHS` are the left and right hand side operands and `OP` encodes the type of the arithmetic operation. In the following we refer to this combination of expression templates and syntax trees as *expression trees*.

As we have seen in (3), the system matrix is build from plugging trial and test functions into the bilinear form. Consequently, we start with the introduction of placeholders for functions in the weak formulation, which have to distinguish

between trial and test functions.

```
1 template <long num, typename diff_tag>
2 struct basisfun;
```

If the template parameter `num` is one, the class is a placeholder for a test function, otherwise it is a placeholder for a trial function. The template parameter `diff_tag` allows to specify derivatives of function for which `basisfun` is a placeholder. The differentiation tag `diff_tag` can also be nested:

```
1 // placeholder for a test function v
2 basisfun<1>
3 basisfun<1, diff<0> > // for dv/dx
4 basisfun<1, diff<1> > // for dv/dy
5
6 //placeholder for d^2 v/dx^2
7 basisfun<1, diff<0,
8         diff<0> >
9         >
```

The latter allows for example to deal with PDEs of fourth order.

A key ingredient in weak formulations are integrations over the full domain, the full boundary or parts of the boundary. Our compile time representation of integrals in the weak formulation is driven by two tag classes [2] that indicate the desired integration domain:

```
1 struct Omega {};
2
3 template <long id>
4 struct Gamma {};
```

The first tag refers to integration over the whole segment and the latter to integration over (parts of) the boundary of the segment. The free template parameter `id` allows to distinguish between several (not necessarily disjoint) subregions of the boundary, where for example Neumann fluxes are prescribed.

The meta class representing integrals takes three template arguments: The domain of integration, the integrand and the type of integration (symbolical or numerical):

```
1 template <typename IntDomain,
2         typename Integrand,
3         typename IntTag>
4 struct IntegrationType;
```

`IntDomain` is one of the two tag classes `Omega` and `Gamma`, `Integrand` is an expression that encodes the integrand, typically of type `Expression`, and `IntTag` is used to select the desired integration method.

After suitable overloads of arithmetic operators acting on `basisfun`, we are ready to specify weak formulations in mnemonic form directly in code. Let us again consider the weak form given in (2). Transferred to code, it reads in two spatial dimensions

```
1 basisfun<1> v;
2 basisfun<1, diff<0> > v_x;
3 basisfun<1, diff<1> > v_y;
4 basisfun<2, diff<0> > u_x;
5 basisfun<2, diff<1> > u_y;
6
7 //the weak formulation:
8 integral<Omega>( u_x*v_x + u_y*v_y )
9 = integral<Omega>( v );
```

Since the gradient in the weak formulation has to be adjusted whenever the spatial dimension of the underlying sim-

ulation domain changes, a convenience class `gradient` was introduced, which represents the mathematical object of a gradient in dependence of the spatial dimension. In principle, `gradient` can be generalized to act on arbitrary arguments and not just on basis functions. Summing up, the full assembly instruction for the weak formulation in (2) on a mesh object `segment` for a matrix `matrix` and a load vector `rhs` can now be written in a single statement in the mnemonic form

```
1 basisfun<1> v;
2 gradient<1, dim> grad_v;
3 gradient<2, dim> grad_u;
4
5 assemble<FEMConfig>(segment, matrix, rhs,
6     integral<Omega>( grad_u * grad_v ) =
7     integral<Omega>( v )
8 );
```

The template parameter `FEMConfig` is a container of type definitions and specifies all FEM related attributes such as the spaces of trial and test functions:

```
1 struct FEMConfig
2 {
3     typedef ScalarTag      ResultDimension;
4     typedef QuadraticBasisfunctionTag
5         TestSpace;
6     typedef QuadraticBasisfunctionTag
7         TrialSpace;
8
9     // further type definitions here
10 };
```

In this way, the specification of details of a particular finite element scheme is separated from the core of linear or linearized finite element iteration schemes, which is to loop over all functions from the test and trial spaces and to generate the system of linear equations from evaluations of the weak formulation at each such function pair. The benefit of this decoupling is that the only necessary change in the code when switching from quadratic to, say, cubic test and trial functions is to modify the two type definitions in `FEMConfig`, all other code remains unchanged.

Another advantage of separate configuration classes such as `FEMConfig` is that one could even switch between different families of discretization schemes. For example, a finite volume discretization could be indicated in another configuration class, e.g. `FVMConfig`. The end-user has to change only one line of code then, while totally different code is generated by the compiler.

The configuration class `FEMConfig` does not contain any information about the spatial dimension and other mesh-related parameters, thus the configuration is effectively independent of the underlying spatial dimension and fully decoupled from any mesh handling. By a highly flexible and clean interface to any mesh-related manipulations we have even managed to use the same code base for arbitrary spatial dimensions, but the discussion of such a domain management is beyond the scope of this paper.

3. POLYNOMIALS AT COMPILE TIME

With the specification of the weak formulation in the previous section, we now proceed with the discussion of test and trial spaces. Typically, these spaces consist of piecewise polynomials defined on a reference element and transformed to the physical elements in space. Therefore, we have implemented compile time representations of polynomials by

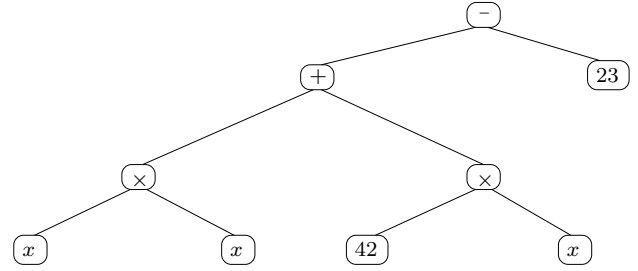


Figure 1: Compile time expression tree for the polynomial $x^2 + 42x - 23$.

reusing the `Expression` class defined in the previous section. A placeholder class `var` for variables like x or y was introduced, taking one integer parameter specifying the index of the unknown the placeholder represents. With suitable overloads of arithmetic and evaluation operators, polynomials can finally be defined and evaluated as

```
1 double result;
2 var<0> x;
3
4 //the polynomial x^2 + 42 x - 23
5 //evaluated at 0.5:
6 result = (x * x + 42 * x - 23)(0.5);
7
8 var<1> y;
9
10 //the polynomial x^2 - xy + y
11 //evaluated at (4.2, 1.3):
12 result = (x * x - x * y + y)(4.2, 1.3);
```

If polynomials are to be evaluated at real-valued arguments, the above code is the best one can get: The polynomials are encoded via template classes at compile time, while the evaluation is carried out at run time. This restriction to evaluation at run time is due to the fact that the present C++ standard does not allow floating point template parameters [8]. However, if the evaluation arguments are known to be integers (and also known at compile time), polynomials can directly be evaluated at compile time using template metaprogramming:

```
1 template <long arg, typename P>
2 double evaluate(P const & polynomial)
3 {
4     return typename EVAL_POLYNOMIAL<P, arg>::
5         ResultType()();
6 }
7
8 void main()
9 {
10     double result;
11     var<0> x;
12
13     //the polynomial x^2 + 42 x - 23
14     //evaluated at 1:
15     result = evaluate<1>(x*x + 42*x - 23);
16 }
```

In contrast to the first code snippet, the expression tree of the polynomial is evaluated by the compiler in the meta-function `EVAL_POLYNOMIAL`. All occurrences of the tag class `var<0>` represented by x in the compile time expression tree are replaced with a wrapper for the scalar value 1, then the resulting expression tree is simplified by removing trivial op-

erations, performing integer operations and the like. In the end, the return statement in the function `evaluate` is optimized by the compiler to `return 20.0;`

In principle it is also possible to allow rational arguments, but due the limited range of integers one is soon confronted with overflows. For example, evaluation of the polynomial x^4 at the fractional 121/1000 leads to a denominator 10^{12} and consequently an overflow. An alternative is to emulate floating point arithmetic at compile time, but the compiler performance was already reported to be atrocious due to the heavy manipulation work [9]. However, direct manipulation of the syntax trees such as replacing all occurrences of `y` with `z` is rather cheap, which is the key ingredient for the remainder of this section.

3.1 Symbolic Differentiation

For the assembly of the system matrix of our model problem (3), derivatives of basis functions (polynomials) are required. In earlier days, these derivatives were computed on the reference element by hand and the result was spread over relevant code lines. Thanks to template metaprogramming and the expression trees introduced in the previous section, the compiler can now compute the required derivatives. All that is left then is to specify the test and trial functions on the reference element.

The differentiation of polynomials is in fact very similar to evaluation. Instead of replacing the placeholder for the unknown with a scalar, we replace the unknown with its derivative, taking the basic rules of differentiation into account:

$$\begin{aligned}(f + g)' &= f' + g' , \\ (f - g)' &= f' - g' , \\ (fg)' &= f'g + fg' , \\ (f/g)' &= (f'g - fg')/g^2 , \\ \partial x_i / \partial x_j &= \delta_{ij}\end{aligned}$$

as well as the fact that derivatives of scalars vanish. Thanks to the functional paradigm of template metaprogramming, the implementation of the metafunction for differentiation is a direct, recursive application of these basic rules. Since the result of the differentiation operation is again an expression tree, we can directly apply the evaluation facilities shown above:

```
1 double result;
2 var<0> x;
3
4 //derivative of x^2 + 42 x - 23
5 //evaluated at 1 during compile time:
6 result = evaluate<1>(differentiate<0>(x*x +
7     42*x - 23));
8
9 var<1> y;
10
11 //derivative w.r.t. y of x^2 - xy + y
12 //evaluated at (4.2, 1.3) during run time:
13 result = differentiate<1>(x * x - x * y + y)
14     (4.2, 1.3);
```

The template argument of `differentiate` denotes the variable as defined by `var`. In the above snippet, 0 corresponds to a differentiation with respect to `x`, while 1 indicates differentiation with respect to `y`. The implementation of the function `differentiate` is similar to that of `evaluate`: It is a convenience

wrapper for a call to the metafunction `DIFFERENTIATE`.

The application of the basic rules of differentiation at compile time may introduce several trivial operations such as multiplications by zero or unity into the compile time expression tree. For compile time evaluations, this is not an issue, but run time evaluations suffer from reduced performance. Consequently, every compile time manipulation is followed by an optimization step, eliminating trivial operations.

3.2 Symbolic Integration

Similar to evaluation and differentiation, the antiderivative of polynomials can also be obtained from compile time manipulations of the underlying expression tree. For the case that the integration bounds are integers, it is also possible to evaluate the antiderivatives at the bounds, hence we are able to compute definite integrals with integer bounds at compile time. Such integer bounds are typically the case for FEM, where reference elements are usually chosen to have corners at points with integer coordinates. For example, the integral

$$\int_0^1 x^2 + 42x - 23 \, dx \quad (4)$$

can be evaluated at compile time by

```
1 double result;
2 var<0> x;
3
4 result = integrate<0,1>(x*x + 42*x - 23);
```

The function `integrate` is a convenience wrapper for the metafunction `INTEGRATE_POLYNOMIAL` and implemented similarly to the function `evaluate` defined previously. Moreover, also nested integration even in case that integral bounds depend on other integration variables have been implemented, which is needed for FEM in higher dimensions. Let us in the following consider the integral

$$\int_0^1 \int_0^{1-x} x(1-x-y)^2 \, dy \, dx , \quad (5)$$

which naturally turns up in higher-order FEM in two spatial dimensions if the reference element is chosen at points (0, 0), (1, 0) and (0, 1).

Our first approach was to carry out the full iterated integration. Each integration consists of the following steps:

- Expand the integrand until it is given as a sum of products of monomials.
- Integrate each summand separately.
- Determine the power of the integration variable in each summand.
- Replace the integration variable by the antiderivative.
- Subtract the term resulting from substituted upper bounds from the term resulting from substituted lower bounds.

Each step was implemented in a separate metafunction. The final iterated integration routine adds these separate metafunctions together and provides the desired functionality. However, one has to expect that the number of summands

in the integrand explodes as the number of integrations increases, especially in the case that integral bounds depend on other integration variables.

To minimize compiler load for the integration over an n -simplex S_n with vertices located at $(0, 0, \dots, 0)$, $(1, 0, \dots, 0)$, $\dots$, $(0, \dots, 0, 1)$, as it is needed for FEM using triangular ($n = 2$) or tetrahedral ($n = 3$) elements, we have first derived the following formula:

$$\int_{S_n} \left(\prod_{i=0}^{n-1} \xi_i^{\alpha_i} \right) \left(1 - \sum_{i=0}^{n-1} \xi_i \right)^{\alpha_n} d\xi = \frac{\alpha_0! \alpha_1! \dots \alpha_n!}{(\alpha_0 + \alpha_1 + \dots + \alpha_n + n)!} \quad (6)$$

This formula allows to avoid any costly iterated integrations, therefore it is sufficient to bring the integrand into the canonical form

$$\sum_k \left(\prod_{i=0}^{n-1} \xi_i^{\alpha_{i,k}} \right) \left(1 - \sum_{i=0}^{n-1} \xi_i \right)^{\alpha_{n,k}} \quad (7)$$

and integrate each summand separately. However, one has to bear in mind that the costly iterated integration is avoided at the cost of fixing the reference element.

Similar to differentiation, an optimization of the transformed expression tree is carried out as a final step. This results in a single rational number for each integral over the reference element and is in terms of efficiency comparable to hard-coding that particular value.

4. COMPUTATION OF ELEMENT MATRICES AT COMPILE TIME

Since the mesh is unknown at compile time, evaluations of the weak form (2) have to be carried out over each cell of the mesh at run time. The standard procedure is to evaluate the transformed weak formulation on a reference element and to transform the result according to the location and orientation of the respective element. This procedure is well described in the literature and makes use of so-called local element matrices [13]. The local element matrix $\mathbf{A}(T)$ for a cell T is typically a linear combination of matrices $\mathbf{A}_k(T_{\text{ref}})$ precomputed on a reference element T_{ref} , thus

$$\mathbf{A}_e(T) = \sum_{k=0}^K \alpha_k(T) \mathbf{A}_k(T_{\text{ref}}), \quad (8)$$

where K and the dimensions and entries of $\mathbf{A}_k(T_{\text{ref}})$ depend on the spatial dimension, the underlying (system of) PDEs and the chosen set of basis functions. While many FEM implementations use hard-coded element matrices, we use the fact that both the weak formulation and the test and trial functions are available at compile time in order to compute these local element matrices during the compilation. At present a compile time integration is supported for simplex cells only, because in that case the Jacobian of the transformation is a scalar and can be pulled out of the resulting integrals.

The transformation of integrals in the weak formulations such as (2) requires the transformation of derivatives according to the chain rule. Thus, this transformation also needs to be applied to the template expression tree as illustrated in Fig. 2 for the case of a product of two derivatives in two dimensions. The class `dt_dx<i,j>` is used to represent the

entries of the Jacobian matrix of the mapping. Since such a transformation is independent from the set of trial and test functions, it has to be carried out only once during compilation, keeping the workload for the compiler low. After expansion of the products and rearrangement, the weak formulation is recast into a form that directly leads to local element matrices as in (8). In a compile time loop the test and trial functions defined on the reference element are then substituted in pairs into this recast weak formulation and the resulting integrals are evaluated symbolically as described in section 3.2. This evaluation has to be carried out for each pair of test and trial functions separately, thus a compile time integration cannot be applied to large sets of test and trial functions without excessive compilation times.

To circumvent the restriction to small sets of test and trial functions for symbolic integration at compile time, our implementation also supports numerical integration. A switch from symbolic to numeric integration is available within the code for the weak formulation:

```

1 // symbolic integration
2 integral<Omega>( grad_u * grad_v ,
3                 AnalyticIntegrationTag() )
4
5 // numerical integration, first order
6 integral<Omega>( grad_u * grad_v ,
7                 LinearIntegrationTag() )
8
9 // default: numerical integration, seventh
10 // order
11 integral<Omega>( grad_u * grad_v )

```

This allows to use several integration rules during the assembly: For integrands which are known to be very smooth, a low order quadrature rule can be assigned, while high order quadrature rules can be applied to less regular integrands.

It has to be emphasized that symbolic integration can only be applied in cases where coefficients in the weak formulation do not show a spatial dependence. For example, the weak form

$$\int_{\Omega} |\mathbf{x}| \nabla u \nabla v \, d\mathbf{x} = \int_{\Omega} |\mathbf{x}| v \, d\mathbf{x} \quad \forall v \in V \quad (9)$$

fails for symbolic integration at compile time due to $|\mathbf{x}|$ in the integrands. Nevertheless, in such a case one has to rely on numerical integration, unless the space dependent part is first projected or interpolated onto polynomials on the reference element.

Hybrid approaches, where integrands without explicit spatial dependence are integrated at compile time and those with spatial dependence are integrated at run time, are also possible. However, they have larger compilation times due to the compile time integration, but hardly improve execution times because most time needs to be spent on the numerical integration anyway.

5. COMPILATION TIMES

We have compared compilation for the assembly of the Poisson equation with weak formulation as in (2) for different polynomial degrees of the trial and test spaces. The benchmarks were carried out using GCC 4.3.2 with optimization flag `-O3` on a machine with a Core 2 Quad 9550 CPU.

Compilation times for full iterated integration, i.e. integrating one variable after another for integrals as in (5), are shown in Tab. 1. In one dimension the numbers stay within an acceptable amount of two minutes. No iterated

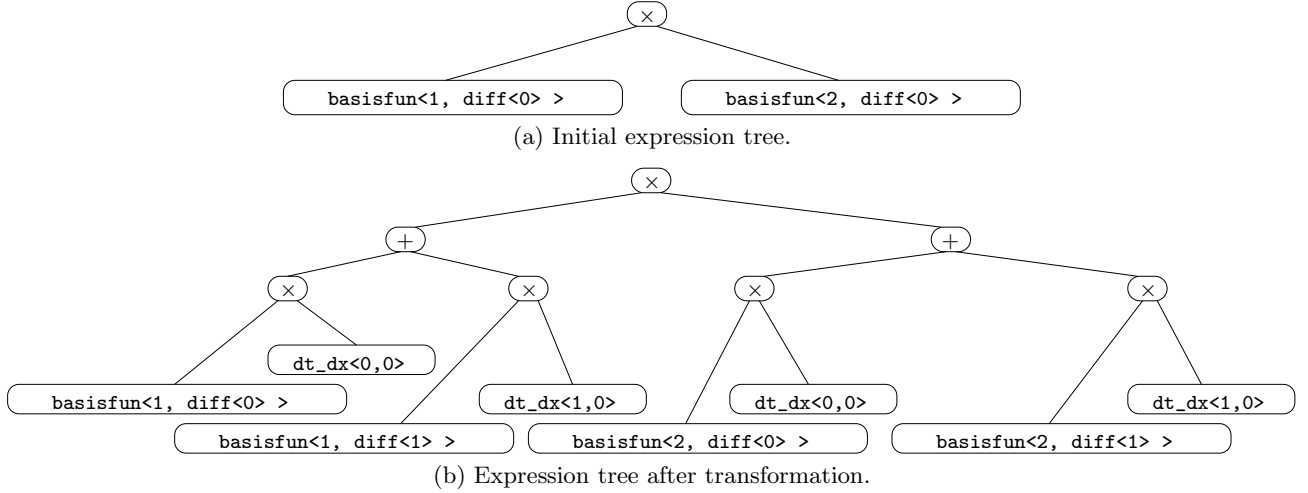


Figure 2: Transformation of the expression tree representing $\partial u/\partial x_0 \times \partial v/\partial x_0$ to a two-dimensional reference element.

	1D	2D	3D
Linear	5s, 321MB	6s, 360MB	11s, 434MB
Quadratic	6s, 341MB	12s, 439MB	126s, 988MB
Cubic	7s, 363MB	384s, 1769MB	-
Quartic	15s, 442MB	-	-
Quintic	86s, 1112MB	-	-

Table 1: Compilation times and compiler memory consumption for several polynomial degrees of the test and trial functions with iterated symbolic integration at compile time in different dimensions. Dashes indicate that the compilation was aborted after ten minutes.

	1D	2D	3D
Linear	5s, 321MB	5s, 329MB	7s, 371MB
Quadratic	5s, 324MB	8s, 375MB	36s, 698MB
Cubic	6s, 326MB	12s, 457MB	424s, 1896MB
Quartic	7s, 328MB	35s, 760MB	-
Quintic	7s, 330MB	148s, 1230MB	-

Table 2: Compilation times and compiler memory consumption for several polynomial degrees of the test and trial functions with formula-assisted symbolic integration at compile time in different dimensions.

integrals have to be computed and the number of test and trial functions increases only linearly with the polynomial order. Nevertheless, more than one gigabyte of memory is required for test and trial functions of order five.

In two dimensions, full iterated integration works up to cubic polynomials, but fails to yield reasonable compilation times and memory requirements for polynomial orders larger than three. The reason for the breakdown is that the number of test and trial functions increases quadratically with the polynomial order and that the integrand gets considerably more complicated due to the polynomials terms.

In three dimensions, triple integrals have to be evaluated on the reference tetrahedron. This increased effort for the compiler leads to reasonable compilation times in the case of linear and quadratic test and trial functions only. Thus, full iterated symbolic integration of element matrices at compile time does not lead to reasonably short compilation times for polynomial order larger than two.

As can be seen in Tab. 2, symbolic integration at compile time using the derived formula (6) leads to reasonable compilation times in one and two dimensions for all test cases. In three dimensions one cannot go beyond cubic basis polynomials for the trial and test spaces without excessive compilation times. The reason is that there are already 20

different cubic test (and trial) functions in three dimensions, so the compiler has to compute 400 entries for each local element matrix. In the case of a polynomial basis of degree four, 35 basis functions require to compute 1225 entries in each local element matrix, which is for current compilers on current desktop computers too much to handle in a reasonable amount of time. A rough extrapolation estimates a compilation time of about 5000 seconds using eight gigabytes of memory for quartic polynomials in three dimensions. Additionally, for more complicated weak formulations, compilation times are further increased due to a larger number of terms in the transformed weak formulation. Nevertheless, due to the often complicated computational domains in real-world applications it is in many cases sufficient to be able to cope with basis polynomials up to third order.

Apart from compilation times there is another limiting factor for symbolic integration: The denominator in the term (6) produces an integer overflow at $13!$, so in three space dimensions with $n = 3$, the criterion

$$\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 < 10 \quad (10)$$

has to be fulfilled. Since the sum of the exponents is roughly twice the polynomial degree of the test and trial functions, one cannot go far beyond degree four even if common factors in the fractional terms are cancelled.

	SI	NI, 1 Point	Exact NI
Linear	0.026	0.025	0.025
Quadratic	0.094	0.105	0.132
Cubic	0.36	0.65	2.17
Quartic	0.96	7.50	88.58
Quintic	1.7	35.9	462

Table 3: Comparison of assembly times (in seconds) for symbolic integration (SI) and numerical integration (NI) for different degrees of basis functions in two dimensions on a triangular mesh with 66049 vertices.

Using numerical integration at run time, but no integration at compile time, the compiler load is much smaller and polynomial orders much larger than three in three dimensions can be handled within less than a minute of compilation times. The drawback of unnecessary numerical integration at run time can be circumvented by a suitable expression engine at run time, as it is implemented e.g. in *Sundance*.

6. EXECUTION TIMES

We have compared execution times for the assembly of the Poisson equation with weak formulation as in (2) for different polynomial degrees of the trial and test spaces. In all our test cases the test space was chosen equal to the trial space and simplex cells were used. The benchmarks were again carried out using *GCC 4.3.2* with optimization flag `-O3` on a machine with a *Core 2 Quad 9550 CPU*.

Matrix access times due to sparse matrix lookup times have been eliminated by redirecting all write operations to a fixed memory position, thus the measured times reflect the time needed to compute the matrix entries, the element transformation coefficients and the lookup times for the indices of the global system matrix. We have compared the symbolic integration with a numerical integration rule using one quadrature point and a quadrature rule with the minimum number of points needed to compute the respective integrals exactly. For polynomials of degree p , we have thus chosen a quadrature rule exact for polynomials up to degree $2p-2$, since according to (2) and (3) each integrand consists of a product of two derivatives of polynomials. The quadrature rule with only one integration point is used to compare the costs of a single evaluation of the integrand relative to other costs.

For a two-dimensional simulation domain with triangular elements, the results in Tab. 3 show that symbolic integration is very attractive for higher order methods. For linear basis functions, there is no notable difference between numerical and symbolic integration. For higher order polynomials we observe that even if only a single quadrature point is used, the increased effort needed to evaluate higher order polynomials leads to a severe difference in execution times of up to a factor of 20 for a quintic basis.

Similar results are obtained in three dimensions, c.f. Tab. 4. A notable difference to the two-dimensional case is that symbolic integration leads to a slightly smaller execution times already in the case of linear polynomials. For higher order polynomials, the number of quadrature points increases as well as the effort needed for each evaluation, leading to

	SI	NI, 1 Point	Exact NI
Linear	0.0064	0.0069	0.0069
Quadratic	0.093	0.120	0.229
Cubic	0.47	0.65	2.82

Table 4: Comparison of assembly times (in seconds) for symbolic integration (SI) and numerical integration (NI) for different degrees of basis functions in three dimensions on a tetrahedral mesh with 4913 vertices.

	Linear	Quadratic	Cubic
Metaprogram. Approach	0.052	0.74	3.78
deal.II 6.1.0 [4]	0.056	1.77	31.20
DOLPHIN 0.9.0 [5]	0.18	1.31	7.16
Getfem++ 3.1 [7]	2.73	8.21	28.37
Sundance 2.3 [10]	0.20	0.53	-
Hand-Tuned Ref.	0.022	0.33	-

Table 5: Execution times (in seconds) for the assembly of the system matrix for the Poisson problem. Linear and quadratic test and trial functions on a tetrahedral mesh with 35937 vertices were compared. Matrix access times are not included.

much larger execution times compared to those obtained with symbolic integration. In the cubic case, the difference is already close to one order of magnitude.

Additionally, we have compared assembly times of our symbolic integration approach with existing FEM software in the case of linear, quadratic and cubic basis polynomials in three dimensions. Again, we have eliminated matrix access times in order to emphasize assembly times. Due to the strongly varying software architectures among the packages, the measured execution times have to be taken with a grain of salt, since other components like for example mesh handling influence the result. The obtained results were compared with a hand-tuned reference implementation that should reflect the achievable performance. The selected packages differ significantly in their architecture: *deal.ii* requires the user to write large parts of the discretization herself. *DOLPHIN* relies on scripts from which C++ code is generated and therefore reflects the family of code generation approaches. *Getfem++* and *Sundance* allow to specify the weak formulation directly in code and parse it at run time.

As can be seen in Tab. 5, our approach leads to good run time efficiency only beaten by *Sundance* in the quadratic case. An interesting observation is the large spread between the execution times, which is more than one order of magnitude compared to the hand-tuned reference implementation. However, especially for simple linear PDEs the assembly times make up only a small amount of the total execution time, which also includes pre- and postprocessing steps and the solution of the resulting linear system. Therefore, differences in execution times for the full solution process show considerably smaller variation among the test candidates.

7. CONCLUSION

We have shown that the application of template metaprogramming together with its functional paradigm in C++ is very well suited for the representation of mathematical objects such as polynomials and operations such as integration or differentiation. The application to FEM allows an abstraction as high as the mathematical formulation so that the weak formulation can directly be transferred from paper to code. Unlike traditional object-oriented programming, template metaprogramming avoids unnecessary dispatches at run time, leading to excellent run time efficiency and short assembly times. Moreover, having the full weak formulation of the underlying mathematical problem available during compile time allows many other optimizations and manipulations at compile time that could have been achieved earlier only by separate, error-prone precompiler. The drawback of our template metaprogramming approach is the longer and memory demanding compilation process, which is still within reasonable limits up to cubic polynomials in three dimensions.

8. REFERENCES

- [1] D. Abrahams and A. Gurtovoy. *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (C++ in Depth Series)*. Addison-Wesley Professional, 2004.
- [2] A. Alexandrescu. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.
- [3] O. Axelsson and V. A. Barker. *Finite Element Solution of Boundary Value Problems: Theory and Computation*. Academic Press, Orlando, Fla., 1984.
- [4] deal.II . Internet: <http://www.dealii.org/>.
- [5] FEniCS project . Internet: <http://www.fenics.org/>.
- [6] freeFEM++ . Internet: <http://www.freefem.org/>.
- [7] Getfem++. Internet: <http://home.gna.org/getfem/>.
- [8] ISO/IEC JTC1 SC22 WG21. The C++ Standard: ISO/IEC 14882:1998, 1998.
- [9] E. Rosten. Floating Point Arithmetic in C++ Templates . Internet: http://mi.eng.cam.ac.uk/~er258/code/fp_template.html.
- [10] Sundance 2.3. Internet: <http://www.math.ttu.edu/~klong/Sundance/html/>.
- [11] D. Vandevoorde and N. M. Josuttis. *C++ Templates*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [12] T. Veldhuizen. Expression templates. *C++ Report*, 7(5):26–31, June 1995.
- [13] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method - Volume 1: The Basis*. Butterworth-Heinemann, 5th edition, 2000.

Fast Multiplication of Large Permutations for Disk, Flash Memory and RAM

Vlad Slavici, Xin Dong*, Daniel Kunkle* and Gene Cooperman*
CCIS Department, Northeastern University, Boston, MA
{vslav,xindong,kunkle,gene}@ccs.neu.edu

ABSTRACT

Permutation multiplication (or permutation composition) is perhaps the simplest of all algorithms in computer science. Yet for large permutations, the standard algorithm is not the fastest for disk or for flash, and surprisingly, it is not even the fastest algorithm for RAM on recent multi-core CPUs. On a recent commodity eight-core machine we demonstrate a novel algorithm that is 50% faster than the traditional algorithm. For larger permutations on flash or disk, the novel algorithm is orders of magnitude faster. A disk-parallel algorithm is demonstrated that can multiply two permutations with 12.8 billion points using 16 parallel local disks of a cluster in under one hour. Such large permutations are important in computational group theory, where they arise as the result of the well-known Todd-Coxeter coset enumeration algorithm. The novel algorithm emphasizes several passes of streaming access to the data instead of the traditional single pass using random access to the data. Similar novel algorithms are presented for permutation inverse and permutation multiplication by an inverse, thus providing a complete library of the underlying permutation operations needed for computations with permutation groups.

Categories and Subject Descriptors

I.1.2 [Symbolic and Algebraic Manipulation]: Algebraic algorithms—*Analysis of algorithms*

General Terms

Algorithms, Experimentation, Performance

Keywords

permutation, permutation multiplication, permutation composition, permutation inverse, pseudo-random permutation

*This work was partially supported by the National Science Foundation under Grant CNS-0916133.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC 2010, 25–28 July 2010, Munich, Germany.

Copyright 2010 ACM 978-1-4503-0150-3/10/0007 ...\$10.00.

1. INTRODUCTION

Algorithms are introduced for efficiently executing the basic permutation operations for large permutations, permutations that range in size from 4 million points to permutations with billions of points.

The standard permutation algorithm is:

```
for  $i \in \{0 \dots N - 1\}$   $Z[i] = Y[X[i]]$ 
```

for input permutation arrays $X[]$ and $Y[]$, and output permutation array $Z[]$. All experiments are performed on random permutations. In this regime, almost every iteration incurs a cache miss.

The size of the permutation dictates the preferred architecture. At the high end of our regime (billions of points), the preferred architecture consists of parallel disks. Using parallel disks, we are able to efficiently multiply permutations with 12.8 billion points in under one hour using the 16 local disks of a 16-node cluster. (Table 4).

In the case of flash memory, it took under one hour to multiply two permutation with 2.5 billion points using a single machine with two solid state flash disks in a RAID configuration (see Table 2).

In the case of RAM, one has a choice of using a multi-threaded algorithm or multiple independent single-threaded processes. Both regimes of computation are useful. Where independent computations from a parameter sweep are performed, or where a parallelization of the higher algorithm is available, independent single-threaded processes are preferred. Where a single inherently sequential algorithm is the goal, the multi-threaded algorithm is preferred.

Experimental results show a 50% speedup in both cases. The novel algorithm has its primary advantage for permutations large enough that they overflow the CPU cache. In the case of a multi-threaded algorithm, we demonstrate the speedup on a recent eight-core commodity computer for permutations with 32 million points (see Table 8). In the case of single-threaded processes, we run eight competing processes simultaneously, and demonstrate the same 50% speedup over the traditional permutation algorithm. In this single-threaded case, the speedup is observed for permutations with as few as 4 million points (see Table 7).

Similar algorithms are also presented for permutation inverse and permutation multiplication by inverse. This completes the standard suite of permutation primitives required by packages that support permutation algorithms, such as GAP [6].

The importance of these new methods for computational group theory is immediately evident by considering a previous permutation computation of one of the authors. In 2003,

a group membership permutation computation for Thompson's group was reported. Thompson's group acts on 143,127,000 points [4]. Those 143 million points from seven years earlier are well within the regime of interest discussed in this paper: between 4 million points and billions of points. That computation now fits on today's commodity computers, including the in-RAM technique of this paper, and would be expected to produce a result 50% faster.

In addition to permutations being given directly, permutations arise frequently as the output of a Todd-Coxeter coset enumeration algorithm. There are several excellent descriptions of this algorithm [1, 5, 13, 17]. In those cases, the first description of the group is as a finite presentation, and one employs coset enumeration to convert this into a more tractable permutation representation. The group can then be efficiently analyzed through such algorithms as Sims's original polynomial-time group membership and the rich library that has grown up around it. Examples of such large coset enumerations include parallel coset enumeration [2] used to find a permutation representation of Lyons's group on 8,835,156 points, sequential coset enumeration [7] used to find a different permutation representation of Lyons's group on 8,835,156 points, and a result [8] finding a permutation representation of Thompson's group on 143,127,000 points.

1.1 Problem Description

In addition to the problem of permutation multiplication, two other standard permutation operations are typically supported by permutation subroutine packages: *permutation inverse* and *permutation multiplication by an inverse*. The last problem, $X^{-1}Y$, is often included as a primitive operation because there exists a more efficient implementation than composing inverse with permutation multiplication:

```
for  $i \in \{0 \dots N - 1\}$   $Z[X[i]] = Y[i]$ 
```

More formally, the problems are:

Let X and Y be two arrays with the same number of elements N , both indexed from 0 to $N - 1$, such that:

$$0 \leq X[i] \leq N - 1, \forall i \in \{0 \dots N - 1\}$$

PROBLEM 1.1 (MULTIPLICATION). *Compute the values of another array, Z , with N elements, defined as follows:*

$$Z[i] = Y[X[i]], \forall i \in \{0 \dots N - 1\}$$

PROBLEM 1.2 (INVERSE). *Compute X^{-1} such that:*

$$X[X^{-1}[i]] = X^{-1}[X[i]] = i, \forall i \in \{0 \dots N - 1\}$$

PROBLEM 1.3 (MULTIPLY BY INVERSE). *Compute the result of multiplying a permutation by an inverse $X^{-1} \times Y$:*

$$Z[i] = Y[X^{-1}[i]], \forall i \in \{0 \dots N - 1\}$$

1.2 Other Problems

While a full discussion is beyond the scope of this paper, we also note that the new algorithms presented for permutation multiplication also apply to object rearrangement:

```
Object  $Z[N]$ ,  $Y[N]$ 
int  $X[N]$ 
for  $i \in \{0 \dots N - 1\}$   $Z[i] = Y[X[i]]$ 
```

When the size of an object remains small compared to the size of a disk block, flash block, or cache line, then the algorithm can be used on disk, flash, or RAM, respectively. Further, the algorithm described here generalizes in an obvious way when Y is near to a permutation, but whose values

may include duplicate entries from $\{0 \dots N - 1\}$, while omitting other entries from $\{0 \dots N - 1\}$.

Terminology.

In this paper we present three permutation multiplication algorithms for architectures with at least two levels of memory, in increasing order of performance: the “*external sort algorithm*”, the “*buckets algorithm*” and the “*implicit indices algorithm*”.

The terminology “*fast-memory/slow-memory*” refers to an algorithm which uses *slow-memory* as the slower, much larger lower-level memory (the one on which the permutation arrays are stored), and *fast-memory* as the faster, much smaller higher-level memory (which cannot hold the entire permutation arrays).

Organization of the Paper.

The rest of the paper is organized as follows: Section 2 presents related work, Sections 3 and 4 present our new fast algorithms, along with some theoretical considerations on their performance. Section 5 presents new fast algorithms for permutation inverse and multiplication by an inverse. Section 6 presents formulas for the optimal running time, under the assumption that the CPU cores are infinitely fast and that the single bus from CPU to RAM is the only bottleneck (or time to access flash memory or disk). Section 7 presents the experimental results, followed by the conclusion in Section 8.

Overview of the Algorithms.

Six algorithms are presented. Algorithms 1 and 2 are intended solely to explore the design space. Algorithms 1 and 2 are disk-based permutation multiplication algorithms using external sorting and a simple buckets technique, respectively. Algorithm 3 reviews an older method for permutation multiplication [3, 4], here called *implicit indices*. Algorithm 4 constitutes the central novelty of this work. It presents a multi-threaded parallel permutation multiplication algorithm. Tables 4 and 5, along with Section 3.2, present a generalization to parallel distributed disks. Algorithms 5 and 6 review older algorithms for permutation inverse and multiplication by inverse [3, 4], that are analogous to Algorithm 3. The generalization to the multi-threaded case (analogous to Algorithm 4) is omitted for lack of space, but experimental results are presented in Table 8.

Section 6 presents a new timing analysis applicable to Algorithms 3, 4, 5 and 6 and their parallel generalizations.

2. RELATED WORK

The current work builds upon [3]. In that work, the authors present a fast RAM-based permutation algorithm that worked well on the Pentium 4, due in part to the 128-byte cache line on that CPUs. Most later CPUs have 64-byte cache lines, and so that algorithm, which is reviewed in this paper as Algorithm 3, later achieved mixed results. Algorithm 3 was also used as a sequential disk-based algorithm in [4]. Related sequential algorithms for permutation inverse and permutation multiplication by inverse were also described in [3, 4].

For lower-level memory data, some of the main ideas of *disk-based computing* [14, 16] have been used successfully in recent years to solve or make progress on important problems

in computational group theory [9, 10, 14, 15], where the size of the data is too large for one RAM subsystem or even the aggregate RAM of a cluster.

The memory gap and memory wall phenomena are very important for understanding the reasons behind the efficiency of our new algorithms and the limitations of both our new algorithms and the traditional algorithms. These phenomena are well-known in literature [3, 18]. All the algorithms we describe, whether traditional or new, are memory-bound for certain parameters.

3. PERMUTATION MULTIPLICATION USING EXTERNAL MEMORY

New algorithms for large permutations are presented. For many problems in computational group theory, the size of a permutation is in the range of tens to hundreds of gigabytes.

The first case presented below deals with permutations that fit on a single disk, with a permutation occupying at least 10 GB of space, but not more than 50 GB. These same algorithms can be run on flash memory. Both disk and flash are types of external memory in wide use today. Table 2 presents experimental results obtained by running our implicit indices algorithm both on flash and on disk. In the following three subsections one can replace disk with flash and everything remains correct.

3.1 Local Disk and Flash

The traditional implementation for permutation multiplication would be:

```
for (i = 0; i < N; i++) Z[i] = Y[X[i]];
```

Using this implementation would be impractical. For large enough pseudo-random permutations, most array accesses are to random locations on disk. Thus a memory page would be swapped in from disk at almost every array element access. On most current systems a memory page is on the order of 4 KB. If the element size is 8 bytes, then for each 8 bytes the traditional algorithm accesses the system would actually transfer 4 KB of data, which results in a 4 KB/8 bytes = 512 times ratio of transferred to useful data. This was indeed observed for naive permutation multiplication running in virtual memory (see Table 3).

A few important notions are defined before discussing the details of the three new algorithms for external memory.

Definition 1. System and Algorithm Parameters

The values in each permutation array X , Y and Z can be represented on β bytes.

$Hlms$ = the size of the higher-level memory component, in number of elements of β bytes.

Any arrays used in the algorithms can be divided into blocks of length $Bl = (Hlms/2)$ number of elements. Two blocks must simultaneously fit in $Hlms$.

$Nb = N/Bl$ is the total number of blocks in an array.

3.1.1 Using External Sort

The disk-based permutation multiplication method using external sorting is described in Algorithm 1.

Using the concept of buckets that fit in RAM, one can significantly improve the performance of the algorithm. RAM buckets are an alternative to external sorting which trades the $n \log n$ running time of sorting for random access within RAM. RAM buckets have significantly sped up computations that previously used external sorting [11].

Algorithm 1 Permutation Multiplication Using External Sort

Input: Permutation arrays X and Y , of size N

Output: Z , s.t. $Z[i] = Y[X[i]]$, $\forall i \in \{0 \dots N-1\}$

Phase 1: Scan X and, for each index i , save the pair $(i, X[i])$ to an array D on disk.

Phase 2: Externally sort all pairs $(i, X[i])$ in array D increasingly by $X[i]$. Now $\forall j \in \{0 \dots N-1\} \exists i \in \{0 \dots N-1\}$ such that $D[j] = (i, X[i])$ and $X[i] = j$.

Phase 3: Scan both array Y and the pairs $(i, X[i])$ in the array D at the same time. $\forall j \in \{0 \dots N-1\}$ we have $D[j] = (i, X[i])$, such that $X[i] = j$. Save the pair $(i, Y[j])$ to an array D' on disk.

Phase 4: Externally sort the array D' increasingly by the index i in pairs $(i, Y[j])$. Now the D' array contains pairs $(i, Y[X[i]])$ in increasing order of i . For each index i , copy $Y[X[i]]$ to the i index in the Z array.

3.1.2 Using RAM Buckets

The RAM buckets method is described in Algorithm 2. The RAM bucket size has to be chosen such that two RAM buckets simultaneously fit in RAM. Considering that both the index i and the value $X[i]$ are represented using the same number of bytes, one needs $2 \times N/Hlms$ buckets (here $Hlms$ is the size of RAM).

Algorithm 2 Permutation Multiplication Using RAM buckets

Input: Permutation arrays X and Y , of size N

Output: Z , s.t. $Z[i] = Y[X[i]]$, $\forall i \in \{0 \dots N-1\}$

1: All arrays are split into Nb equally sized buckets, each containing $Bl = N/Nb$ elements. The bucket size can be at most one-half the size of RAM. Bucket i of array A is denoted A_i . Bucket b contains indices in the range $[b * Bl, (b + 1) * Bl)$.

// **Phase 1: bucketize**

2: Scan array X and, for each index i , save the pair $(i, X[i])$ in the bucket $D_{X[i]/Bl}$.

// **Phase 2: permute buckets**

3: **for** each bucket b **do**

4: Load buckets D_b and Y_b into RAM.

5: **for** each index i in this bucket **do**

6: Let $D_b[i] = (j, X[j])$.

7: Save the pair $(j, Y_b[X[j]])$ to bucket $D'_{j/Bl}$.

// **Phase 3: combine buckets**

8: **for** each bucket b **do**

9: Load buckets D'_b and Z_b into RAM.

10: **for** each index i in this bucket **do**

11: Let $D'_b[i] = (j, Y[X[j]])$.

12: Set $Z_b[j] = Y[X[j]]$.

Algorithm 2 presents a few important improvements over Algorithm 1. Note that in phase 2 of Algorithm 2, there is no need to save the index in the buckets of array Y , since it is implicit in the ordering. Thus a bucket of array Y occupies twice as little space as a bucket of pairs $(i, X[i])$. In phase 3, Z is also divided into $2 \times N/Hlms - 1$ buckets, and all indices from the j -th bucket of D' correspond to positions in the j -th bucket of Z . Algorithm 2 completely eliminates sorting and, in practice, shows a 4 times (or more) speedup over the External Sort-based algorithm if the computation is disk-bound (see Table 5, the 1 node case).

Both algorithms 1 and 2 need to save the index of each value of the X permutation, thus resulting in disk arrays as large as twice the size of the initial arrays. The implicit indices RAM/disk algorithm (Algorithm 3) avoids saving the indices to disk arrays.

3.1.3 With Implicit Indices

Algorithm 3 Permutation Multiplication using implicit indices

Input: Permutation arrays X and Y , of size N

Output: Z , s.t. $Z[i] = Y[X[i]]$, $\forall i \in \{0 \dots N-1\}$

- 1: All arrays are split into Nb equally sized buckets, each containing $Bl = N/Nb$ elements. The bucket size can be at most one-half the size of RAM. Bucket i of array A is denoted A_i . Bucket b contains indices in the range $[b * Bl, (b + 1) * Bl)$.
// Phase 1: bucketize
 - 2: Traverse the X array and distribute each value $X[i]$ into bucket $D_{X[i]/Bl}$ on disk.
// Phase 2: permute buckets
 - 3: **for** each bucket b **do**
 - 4: Load buckets D_b and Y_b into RAM.
 - 5: **for** each index i in this bucket **do**
 - 6: Set $D_b[i] = Y_b[D_b[i]]$.
// Phase 3: combine buckets
 - 7: For each value $X[i]$, let j be the next value in bucket $D_{X[i]/Bl}$. Note that $j = Y[X[i]]$. Set $Z[i] = j$ and remove that value from bucket $D_{X[i]/Bl}$.
-

The correctness of Algorithm 3 can be proved by following the three phases for a generic index $i \in \{0 \dots N-1\}$: in phase 1 value $X[i]$ is distributed into bucket $j = X[i]/Bl$ at position k of array D , so that $D[k] = X[i]$. In phase 2, $D[k] = Y[D[k]]$, which can be written $D[k] = Y[X[i]]$. In phase 3, $Z[i] = D[k]$, which can be written $Z[i] = Y[X[i]]$.

The implicit indices version runs about twice as fast as the buckets version (see Table 4). The implicit indices RAM/disk algorithm performs the following steps: a sequential read of the X array and a sequential write of the D (temporary) array in phase 1 (*2 sequential accesses*); a sequential read of the D array, a sequential read of the Y array and a sequential write of the D array (*3 sequential accesses*); and a sequential read of the X array, a sequential read of the D array and a sequential write of the Z array (*3 sequential accesses*). In total, there are 8 sequential accesses

It is interesting to compare the running time of the implicit indices algorithm and the running time of a permutation multiplication algorithm that we implemented in Roomy [12], which uses Algorithm 2. Roomy is a general framework for disk-based and parallel disk-based computing which provides a high-level API for manipulating large amounts of data. The disk-based implicit indices algorithm is generally twice as fast as the Roomy implementation.

3.2 Many Disks

Here we describe how the three disk-based algorithms for permutation multiplication, presented in Section 3.1, can be used with the many disks in a cluster of computers.

Serial permutation multiplication using external sort is described in Algorithm 1. To parallelize it, all arrays are first

split into sub-arrays, each of which is placed on the disk of a single compute node in the cluster. All operations on those arrays are performed in parallel. In cases where one node generates data that references a sub-array on another node, that data is first sent over the network, then saved to disk. In our implementation, there is a separate thread of execution on each node that handles the writing of this remote data to the local disk. Finally, there is a synchronization point after each phase, to insure that all nodes are done with one phase before beginning the next.

Permutation multiplication using buckets (Algorithm 2) is made parallel in the same way. The arrays are already split into sub-arrays (buckets), and the same methods are used for data distribution, parallel processing, and synchronization.

There is one additional modification necessary to parallelize permutation multiplication using implicit indices (Algorithm 3). Because the algorithm depends on the specific ordering of elements in each bucket, the buckets can not be written to in parallel. This is solved in the same way that Algorithm 4 extends Algorithm 3: each bucket is further split into sub-buckets, so that each node has its own sub-bucket to write to. Unlike the multi-threaded RAM case, the parallel disk case does not need an extra phase to compute the sizes of the sub-buckets, since the buckets are represented with files, which are dynamically sized.

4. PERMUTATION MULTIPLICATION IN RAM

The traditional permutation multiplication algorithm for cache/RAM can be trivially-parallelized. Each thread processes a contiguous region of the $X[]$ permutation array. Although this incurs frequent cache misses, it tends to scale linearly on current commodity computers *until one goes beyond four cores*. This is because the single bus to RAM becomes saturated by the pressure of the several cores. In Table 8 of Section 7, one sees this happening approximately with 3 threads for permutation multiplication and for 4 threads for inverse and multiplication by inverse.

Algorithm 3 of Section 3 presented a single-threaded disk-based algorithm to overcome the many page faults. The same algorithm can be implemented for cache/RAM to minimize cache misses. That algorithm's cache/RAM version is preferred for permutation algorithms that can be parallelized at a higher level and then call a single-thread permutation multiplication algorithm. Here, we consider a multi-threaded version for the case when the higher level algorithm does not parallelize well. The corresponding results at the level of eight cores are presented in Table 7 in Section 7. As described in the extrapolation in Section 7.3, both the new single-threaded and the new multi-threaded algorithms are expected to have an even greater advantage at the 16-core and higher level in the future.

Algorithm 4 provides the multi-threaded version for multiplication using cache/RAM. Intuitively, it operates by splitting the buckets of Algorithm 3 into sub-buckets. Within a given bucket, each thread "owns" a contiguous region (a sub-bucket) for which it has responsibility. Algorithm 4 requires one extra phase (Phase 1) in order to determine in advance the size of the sub-bucket to allocate for each thread.

Some alternative designs were also explored. A brief summary of the alternatives considered is presented along with our reasons for rejecting them.

Algorithm 4 Multi-threaded cache/RAM Permutation Multiplication using Implicit Indices

Input: Permutation arrays X and Y , of size N , the number of cache buckets Nb , the number of threads T .

Output: Z , s.t. $Z[i] = Y[X[i]]$, $\forall i \in \{0 \dots N-1\}$

- 1: All arrays are split into Nb equally sized buckets, each containing $Bl = N/Nb$ elements. The bucket size can be at most one-half the size of cache. Bucket i of array A is denoted A_i . Bucket b contains indices in the range $b \times Bl$ to $(b+1) \times Bl$.
 - 2: Each thread t , $0 \leq t \leq T-1$, will handle indices in the range $t \times N/T$ to $(t+1) \times N/T - 1$.
// Phase 1: create sub-buckets
 - 3: Create a temporary array D , split into $T \times Nb$ sub-buckets. $D_{b,t}$ is the sub-bucket corresponding to bucket b and thread t . The bucket D_b is the concatenation of all sub-buckets $D_{b,t}$. The size of a sub-bucket is first determined by an additional scan of X .
// Phase 2: bucketize
 - 4: Each thread scans the portion of X that it is responsible for, and saves each $X[i]$ to sub-bucket $D_{t,X[i]/Bl}$.
// Phase 3: permute buckets
 - 5: Each thread locally permutes each bucket b that it is responsible for, setting $D_b[i] = Y_b[D_b[i]]$.
// Phase 4: combine buckets
 - 6: Each thread computes the final values $Z[i]$ that it is responsible for. For each such index i , let j be the next value in sub-bucket $D_{t,X[i]/Bl}$ that has not been removed (Note that $j = Y[X[i]]$). Set $Z[i] = j$ and remove that value from sub-bucket $D_{t,X[i]/Bl}$.
-

- Using pthread private data via “_thread” (problem: uses too much memory).
- Using pthread locks to synchronize memory access (problem: synchronization delays).
- Using an atomic add operation to a single global counter (problem: internally, it still uses a lock).
- Exploiting L1 cache via a two-level algorithm, similar to two-level external sort (problem: delays due to extra passes).

Section 7.3 presents experimental results for the cache/RAM multi-threaded implicit indices algorithm.

5. PERMUTATION INVERSE. MULTIPLICATION BY AN INVERSE

While Algorithms 5 and 6 are not new [3, 4], their multi-threaded generalizations analogous to Algorithm 4 are novel. Experimental results for running permutation inverse and multiplication by an inverse, as well as theoretical estimates for these runs, can be found in Table 8.

Permutation Inverse.

The traditional algorithm for permutation inverse is:

```
for (i = 0; i < N; i++) Y[X[i]] = i;
```

The bottleneck is still the random access (this time write access) to the Y array.

Algorithm 5 Permutation Inverse Using Implicit Indices

Input: Permutation array X , of size N

Output: $Y[X[i]] = i$, $\forall i \in \{0 \dots N-1\}$

Phase 1: Scan array X and distribute each value $X[j]$ in array D at block number $k = X[j]/Bl$. At the same time write value j at the same index in block k of D' as $X[j]$ was written at in block k of D .

Phase 2: Scan the D' and D arrays sequentially at the same time and, for each index j , write $Y[D[j]] = D'[j]$.

Permutation Multiplication by an Inverse.

For the multiply-by-inverse, the traditional algorithm is:

```
for (i = 0; i < N; i++) Z[X[i]] = Y[i];
```

At the end of the loop $Z[i] = Y[X^{-1}[i]]$, $\forall i \in \{0 \dots N-1\}$.

Algorithm 6 Permutation Multiplication by an Inverse Using Implicit Indices

Input: Permutation arrays X and Y , of size N

Output: $Z[i] = Y[X^{-1}[i]]$, $\forall i \in \{0 \dots N-1\}$

Phase 1: Scan array X and distribute each value $X[j]$ in its corresponding block of array D . At the same time write value $Y[j]$ at the same index in D' as $X[j]$ was written at in D .

Phase 2: Scan the D' and D arrays sequentially and, for each index j , write $Y[D[j]] = D'[j]$.

6. PERFORMANCE ANALYSIS

The analysis presented here can be used to estimate the running time for the implicit indices algorithms, when using any 2-level memory hierarchy, including cache/RAM, RAM/flash, RAM/disk. The implicit indices algorithms include Algorithm 3, its generalization to Algorithm 4, Algorithms 5 and 6, and their parallel generalizations.

Definition 2. System and Algorithm parameters (Analysis)

Hrl = higher-level read memory latency (*seconds*)

Lrl = lower-level read memory latency (*seconds*)

Lwl = lower-level write memory latency (*seconds*)

Lwb = lower-level write memory bandwidth (*bytes/second*)

Lrb = lower-level read memory bandwidth (*bytes/second*)

Es = array-element size (*bytes*)

N = array length (*bytes*)

Nb = number of blocks per array

Bs = bucket size (*bytes*)

$Bl = N/Nb$ (block length) (*bytes*)

We refer to permutation multiplication as PM, to permutation inverse as PI, and to permutation multiplication by an inverse as PMI. The next three formulas estimate the running time when memory is the bottleneck for PM, PI, and PMI, respectively. Note that in the case of cache/RAM N/Lrb must be added to each formula, due to the extra pass.

FORMULA 6.1 (PM TOTAL ESTIMATED TIME).

$$N \times \left(\frac{Lwl + Lrl}{Bs} + \frac{3}{Lwb} + \frac{5}{Lrb} + \frac{Hrl}{Es} \right)$$

FORMULA 6.2 (PI TOTAL ESTIMATED TIME).

$$3N \times \left(\frac{1}{Lrb} + \frac{1}{Lwb} + \frac{2 \times Lwl}{Bs} + \frac{Hrl}{Es} \right)$$

FORMULA 6.3 (PMI TOTAL ESTIMATED TIME).

$$N \times \left(\frac{4}{Lrb} + \frac{3}{Lwb} + \frac{2 \times Lwl}{Bs} + \frac{Hrl}{Es} \right)$$

7. EXPERIMENTAL RESULTS

7.1 Local disk and flash

Tests were ran on an AMD Phenom 9550 Quad-Core at 2.2 GHz with 4 GB of RAM, running Fedora Linux with kernel version 2.6.29. The machine has both a disk drive (Seagate Barracuda 7200.10 250GB) and 2 RAID-ed flash SSD drives ($2 \times$ INTEL SSD SSDSA2MH080G1GC, 80 GB each).

Table 1 contains the measured system parameters of this machine. Table 1 also contains the measured system parameters for one of the disks of the cluster that was used to run the “parallel RAM/parallel disk” algorithms. The parallel disk bandwidth assumes that network bandwidth is not a limiting factor. Table 4 shows this to be the case for permutation arrays of size up to 25 GB.

Table 1: Measured system parameters for external memory.

	Disk	Flash	Cluster disk
Read BW (MB/s)	85	200	51
Write BW (MB/s)	82	26	51
Latency (ms)	10	14	39
Latency RAM (ns)	233	211	169

Table 2 shows a comparison between the new RAM/disk algorithm and the new RAM/flash algorithm, both based on implicit indices. The estimates from the formulas of Section 6 are also presented, to confirm that the algorithm is limited by the bandwidth of disk and flash.

Table 2: Running times of our new RAM/disk and RAM/flash algorithms and comparison with estimated running times. Element size is 8 bytes. Bucket size is 2 MB, block size is 1 GB.

Nr. elts. (billions)	Running Time (seconds)					
	Using Disk					
	PM		PI		PMI	
	real	est	real	est	real	est
1.25 (10 GB)	1609	1388	1002	1149	1253	1269
2.5 (20 GB)	3205	2776	2259	2298	2736	2538
	Using flash					
	PM		PI		PMI	
	real	est	real	est	real	est
1.25 (10 GB)	1584	1849	1212	1747	1348	1798
2.5 (20 GB)	2807	3698	2604	3494	2711	3596

Table 3 details our findings about the traditional permutation multiplication algorithm ran in virtual memory on the same machine. The experimental results confirmed our expectations: when the working set is at least twice the size of available RAM, using the traditional algorithm in virtual

Table 3: Comparison of the traditional algorithm and the buffered traditional algorithm with disk-based and flash-based external memory. Element size: 4 bytes. RAM size is 4 GB. Arrays X, Y and Z are the work set.

Nr. elts (millions)	traditional algorithm time (seconds)			
	sequential		parallel	
	disk	flash	disk	flash
750 (3.0 GB)	3476	1198	1802	489
825 (3.5 GB)	> 4hrs	> 4hrs	> 4hrs	> 4hrs
	Buffered algorithm time (seconds)			
	sequential		parallel	
	disk	flash	disk	flash
750 (3.0 GB)	150	130	142	115
825 (3.5 GB)	> 4hrs	11762	> 4hrs	3561

memory is infeasible. We also implemented a buffered traditional algorithm and ran parallel versions of both the simple traditional and buffered traditional algorithm. While the parallel buffered traditional algorithm clearly outperforms the parallel simple traditional one, the first is still infeasible when the working set overflows RAM by a significant percentage.

7.2 Many disks

These experiments were run on a cluster of computers, each with two dual-core 2.0 GHz Intel Xeon 5130 CPUs and 16 GB of RAM, a locally attached 500 GB disk, running Linux kernel version 2.6.9. The network used a Dell PowerConnect 3348 Fast Ethernet switch. Only one process was used per node, to avoid competition for the single disk.

Tables 4 and 5 give a comparison of the three disk-based permutation algorithms presented in Section 3.1, based on: external sorting; RAM buckets; and implicit indices.

Table 4: Comparison of three parallel-disk permutation multiplication algorithms for increasing permutation size, using 16 nodes of a cluster. Elements are 8 bytes each. A “*” indicates that the estimated time is not accurate, because the network became a bottleneck.

Nr. elts. (billions)	Algorithm Time (seconds)			
	Sort	Bucket	Implicit Indices	
			real	estimated
0.8 (6 GB)	538	105	77	70
1.6 (12 GB)	1151	202	100	139
3.2 (24 GB)	3440	490	270	279
6.4 (48 GB)	7484	2364	1571	*
12.8 (95 GB)	15697	6838	3228	*

Table 4 shows the results of using 16 nodes of a cluster, with permutation sizes ranging from 800 million elements (6 GB) to 12.8 billion elements (95 GB). In general, the three algorithms scale roughly linearly with permutation size. The most notable exception is a 5-fold increase in the running times of the bucket and implicit indices algorithms when

Table 5: Comparison of three parallel-disk permutation multiplication algorithms for increasing parallelism, using from 1 to 16 nodes of a cluster. Elements are 8 bytes each. Permutations have 1.6 billions elements each (12 GB).

Nr. nodes	Algorithm Time (seconds)		
	Sort	Bucket	Implicit Indices
1	28952	7069	5576
2	13555	3627	2861
4	6197	677	354
8	2227	336	167
16	1185	202	100

moving from 24 GB to 48 GB permutations. We believe that this is due to network traffic on an older Fast Ethernet switch. Until that point, the bottleneck was likely disk bandwidth. The sorting based algorithm does not see a similar effect because its time is dominated by the in-RAM sorting process, not inter-node communications.

Table 5 shows the results of using between 1 and 16 nodes of the cluster, with permutations having 1.6 billion elements (12 GB). Again, the time for each algorithm scales roughly linearly with the number of nodes. The non-linear scaling when moving from 2 to 4 nodes is likely due to the bottleneck moving between disk and the network.

In general, the bucket algorithm takes about 1.5 to 2 times longer than the implicit indices algorithm, with the largest differences occurring with larger permutations and more parallelism. The implicit indices algorithm is more efficient because of the smaller amount of data that must be saved to disk. The sorting based algorithm takes roughly 5 to 10 times longer than the implicit indices algorithm, largely due to the time needed to sort data in RAM.

7.3 RAM

For cache/RAM, the performance of permutation multiplication, inverse and multiplication by an inverse was demonstrated on a recent 8-core commodity machine: two Quad-core Intel Xeon E5410 CPUs running at 2.33 GHz, with a total of 24 MB L2 cache — 12 MB L2 cache per socket and 16 GB of RAM made up of four memory modules. Table 6 lists the system parameters measured on this system.

Table 7 concerns the case of independent permutation computations running in parallel, with one computation per core. We believe that the traditional algorithm is close to saturating the bandwidth from CPU to RAM, both in the case of 8 threads and 8 processes. Table 8 provides confirming evidence of bandwidth saturation in comparing 4 threads versus 8 threads. As described in Section 6, the new algorithm is more bandwidth-efficient. We see that benefit for 8 processes but not for 8 threads. We speculate that is due to cache poisoning as the threads compete for the same cache.

Table 6: Measured system parameters for cache/RAM. Latency for cache is negligible.

Read bandwidth	5859 MB/s
Write bandwidth	3850 MB/s
Latency of 1 random access	302 ns

In Table 8 one can find running times for Algorithm 4 and

Table 7: Comparison of traditional and new algorithms, using thread or process-based parallelism. Permutations have 4 million 4-byte elements each.

	Eight Threads	Eight Processes
Traditional	0.042 s	0.048 s
New	0.054 s	0.026 s

the multi-threaded generalizations of Algorithms 5 and 6, as well as theoretical estimates of these running times based on the formulas in Section 6.

The new permutation multiplication algorithm is faster by about 50% than the traditional algorithm for permutations of 32 million elements or more, when using 8 threads. Our new algorithm is also faster than performing 8 multi-threaded traditional permutation multiplications in a row by at least a factor of 1.6. In contrast, when using only one thread (with seven cores idle), the time represents a mixture of RAM bandwidth and CPU power. Hence, the traditional and new algorithms have similar performance.

Extrapolation on memory bandwidth results.

In the near future, commodity machines will continue to gain additional CPU cores at a rate based on Moore’s Law. But the number of memory modules on the motherboard is likely to remain fixed (while the density of each memory module continues to rise). Hence the memory bandwidth is unlikely to grow significantly.

Table 8 shows the times for the traditional algorithm already approaching an asymptotic value for the transition from 4 threads to 8 threads. Furthermore, the timings for 8 threads is close to the timing for the theoretically optimal case for bandwidth limited computation.

The new algorithm shows a significant improvement in time in the transition from 4 threads to 8 threads. In the case of permutation multiplication, the timing for 8 threads approaches that of the theoretically optimal memory bandwidth limited case. On the other hand, the algorithms for permutation inverse and permutation multiplication by an inverse show the potential for additional improvements in timings as more cores become available. This is seen by comparing the numbers for 8 threads and the optimal case.

8. CONCLUSIONS

New algorithms were presented for multiplication of large permutations for disk and flash (Section 3), for the aggregate disks of a cluster (Section 3.2) and a multi-threaded algorithm for RAM (Section 4). These algorithms make permutation multiplication a practical operation for large permutations that do not fit in RAM. Further, the multi-threaded cache/RAM implicit indices algorithm clearly outperforms the trivially-parallel traditional algorithm when using multiple threads on machines with many cores.

Acknowledgments.

We gratefully acknowledge CERN for making available an 8 core machine for testing.

9. REFERENCES

- [1] J. J. Cannon, L. A. Dimino, G. Havas, and J. M. Watson. Implementation and analysis of the

Table 8: Running times (seconds) of our new implicit indices permutation multiplication for cache/RAM. As explained, we need a machine with at least 8 cores working on the new algorithm in parallel for the CPU to be a less significant factor. Element size is 4 bytes. A bucket here is a cache line, the block size is variable between runs. The values in the column labeled “Optimal” are derived from the equations in Section 6 using values based on Table 6.

Nr. elem. (millions)	Running Time (seconds)									
	1 thread		2 threads		4 threads		8 threads		Optimal	
32 (128 MB)	Permutation Multiplication									
	new	trad.	new	trad.	new	trad.	new	trad.	new	trad.
	0.81	0.81	0.70	0.51	0.43	0.45	0.29	0.42	0.25	0.39
	64 (256 MB)	1.98	1.67	1.47	1.27	0.85	0.95	0.62	0.88	0.50
128 (512 MB)	4.68	4.09	2.98	2.52	1.72	2.10	1.16	1.81	1.01	1.54
32 (128 MB)	Permutation Inverse									
	new	trad.	new	trad.	new	trad.	new	trad.	new	trad.
	1.03	1.83	0.86	1.04	0.54	0.63	0.34	0.59	0.18	0.53
	64 (256 MB)	2.39	3.70	1.75	2.06	0.96	1.31	0.66	1.21	0.37
128 (512 MB)	5.33	7.48	3.52	4.15	2.00	2.65	1.35	2.51	0.75	2.11
32 (128 MB)	Permutation Multiplication by an Inverse									
	new	trad.	new	trad.	new	trad.	new	trad.	new	trad.
	1.06	1.84	0.87	1.10	0.53	0.66	0.35	0.60	0.20	0.55
	64 (256 MB)	3.72	2.46	1.77	2.24	0.99	1.33	0.70	1.23	0.41
128 (512 MB)	5.57	7.52	3.62	4.22	2.08	2.72	1.43	2.55	0.83	2.19

- Todd-Coxeter algorithm. *Math. Comp.*, 27:463–490, 1973.
- [2] G. Cooperman and G. Havas. Practical parallel coset enumeration. In *Proc. of Workshop on High Performance Computation and Gigabit Local Area Networks*, volume 226 of *Lecture Notes in Control and Information Sciences*, pages 15–27. Springer Verlag, 1997.
- [3] G. Cooperman and X. Ma. Overcoming the memory wall in symbolic algebra: A faster permutation algorithm (formally reviewed communication). *SIGSAM Bulletin*, 36:1–4, Dec. 2002.
- [4] G. Cooperman and E. Robinson. Memory-based and disk-based algorithms for very high degree permutation groups. In *ISSAC*, pages 66–73, 2003.
- [5] H. Felsch. Programmierung der Restklassenabzählung einer Gruppe nach Untergruppen. *Numerische Mathematik*, 3:250–256, 1961.
- [6] GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.2*. (<http://www.gap-system.org>), 2000.
- [7] G. Havas and C. Sims. A presentation for the Lyons simple group. In *Computational Methods for Representations of Groups and Algebras*, volume 173 of *Progress in Mathematics*, pages 241–249, 1999.
- [8] G. Havas, L. Soicher, and R. Wilson. A presentation for the Thompson sporadic simple group. In *Groups and Computation III, Computational Methods for Representations of Groups and Algebras*, volume 8 of *Ohio State University Mathematical Research Institute Publications*, pages 193–200. de Gruyter, 2001.
- [9] D. Kunkle and G. Cooperman. Twenty-six moves suffice for Rubik’s cube. In *Proc. of International Symposium on Symbolic and Algebraic Computation (ISSAC ’07)*, pages 235–242. ACM Press, 2007.
- [10] D. Kunkle and G. Cooperman. Solving Rubik’s cube: Disk is the new RAM. *ACM Communications*, 51:31–33, 2008.
- [11] D. Kunkle and G. Cooperman. Harnessing parallel disks to solve Rubik’s cube. *Journal of Symbolic Computation*, 44:872–890, 2009.
- [12] D. Kunkle and G. Cooperman. Roomy. URL: <http://sourceforge.net/apps/trac/roomy/wiki>, 2009.
- [13] J. Neubüser. An elementary introduction to coset table methods in computational group theory. In C. Campbell and E. Robertson, editors, *Groups – St Andrews 1981*, volume 71 of *London Math. Soc. Lecture Note Ser.*, pages 1–45, Cambridge, 1982. Cambridge University Press.
- [14] E. Robinson and G. Cooperman. A parallel architecture for disk-based computing over the baby monster and other large finite simple groups. In *Proc. of International Symposium on Symbolic and Algebraic Computation (ISSAC ’06)*, pages 298–305. ACM Press, 2006.
- [15] E. Robinson, G. Cooperman, and J. Müller. A disk-based parallel implementation for direct condensation of large permutation modules. In *Proc. of International Symposium on Symbolic and Algebraic Computation (ISSAC ’07)*, pages 315–322. ACM Press, 2007.
- [16] E. Robinson, D. Kunkle, and G. Cooperman. A comparative analysis of parallel disk-based methods for enumerating implicit graphs. In *Parallel Symbolic Computation (PASCO ’07)*, pages 78–87. ACM Press, 2007.
- [17] J. Todd and H. Coxeter. A practical method for enumerating cosets of a finite abstract group. *Proc. Edinburgh Math. Soc., II. Ser.* 5, 5:26–34, 1936.
- [18] W. Wulf and S. McKee. Hitting the memory wall: Implications of the obvious. *ACM Computer Architecture News*, 23(1):20–24, 1995.

Author Index

Abramov, Sergei	311	Hubert, Evelyne	1	Roune, Bjarke Hammersholt	115
Al Zain, Abdallah	339	Hutton, Sharon	227	Rump, Siegfried M.	3
Avendaño, Martín	331	Ibrahim, Ashraf	331	Rupp, Karl	347
Barkatou, Moulay A.	7, 45	Jeannerod, Claude-Pierre	281	Rusek, Korben	331
Berkesch, Christine	99	Joldeş, Mioara	147	Safey El Din, Mohab	107, 257
Bodrato, Marco	273	Kaltofen, Erich	227	Schneider, Carsten	211
Bostan, Alin	203	Kapur, Deepak	29	Schwarz, Fritz	93
Brisebarre, Nicolas	147	Kauers, Manuel	195, 211	Sevilla, David	163
Brown, Christopher	69	Khonji, Majid	265	Shi, Xiaoran	171
Cha, Yongjae	303	Konovalov, Alexander	339	Slavici, Vlad	355
Chen, Changbo	187	Kunkle, Daniel	355	Sottile, Frank	179
Chen, Falai	171	Lemaire, François	85	Spaenlehauer, Pierre-Jean	257
Chen, Shaoshi	203	Levy, Giles	297, 303	Stalinski, Thomas	265
Chyzak, Frédéric	203	Leykin, Anton	99	Strzeboński, Adam	61, 69
Conti, Costanza	251	Li, Zijia	155	Sturm, Thomas	77
Cooperman, Gene	355	Li, Ziming	203	Sun, Yao	29
Davenport, James H.	187	Linton, Steve	339	Tiwari, Ashish	5
Dong, Xin	355	May, John P.	187	Trinder, Phil	339
Eberly, Wayne	289	Mayr, Ernst W.	21	Tsarev, Sergey P.	11
El Bacha, Carole	45	Mezzarobba, Marc	139	Tsigaridas, Elias	235, 243
Emiris, Ioannis Z.	235, 243	Moreno Maza, Marc	187	Ürgüplü, Ash	85
Faugère, Jean-Charles	131, 257	Mouilleron, Christophe	281	Vakil, Ravi	179
Galligo, André	235	Mourrain, Bernard	243	Vershelde, Jan	179
Gao, Shuhong	13	Pan, Victor Y.	219	Volny, Frank	13
von zur Gathen, Joachim	123, 131	Pernet, Clément	265	Wachsmuth, Daniel	163
Gemignani, Luca	251	Perret, Ludovic	131	Wang, Dingkang	29
Gerdt, Vladimir	53	Pflügel, Eckhard	45	Xia, Bican	187
Gerhard, Jürgen	9	Pillwein, Veronika	195	Xiao, Rong	187
Giesbrecht, Mark	123	Ritscher, Stephan	21	Yang, Zhengfeng	155
Grigoriev, Dima	93	Robertz, Daniel	53	Yuan, Quan	37
Guan, Yinhua	13	Roch, Jean-Louis	265	Zanoni, Alberto	319
Guo, Feng	107	Roche, Daniel S.	325	Zengler, Christoph	77
Hammond, Kevin	339	Roche, Thomas	265	Zheng, Ai-Long	219
Harvey, David	325	Rojas, J. Maurice	331	Zhi, Lihong	107, 155, 227
van Hoeij, Mark	37, 297, 303	Romani, Lucia	251	Ziegler, Konstantin	123
Horn, Peter	339	Roozmond, Dan	339		