The Joint Conference of ASCM 2009 and MACIS 2009

Asian Symposium on Computer Mathematics

Mathematical Aspects of Computer and Information Sciences

Editors: Masakazu Suzuki, Hoon Hong, Hirokazu Anai,
Chee Yap, Yousuke Sato, Hiroshi Yoshida

About MI Lecture Note Series

The Math-for-Industry (MI) Lecture Note Series is a successor to the COE Lecture Notes, published for the 21st COE Program "Development of Dynamic Mathematics with High Functionality", sponsored by Ministry of Education, Culture, Sports, Science and technology-Japan (MEXT) (From 2003 to 2007).

The MI series reports lectures given by scholars invited under the following two programs: "Training Program of Ph.D. and new Master's in Mathematics as Required by Industry", adopted as a Support Program for Improving Gradute School Education by MEXT (from 2007 to 2009); and "Education-and-Research Hub for Mathematics-for-Industry", newly adopted as a Global COE Program by MEXT (from 2008 to 2012).

July 2008
Masato Wakayama
Global COE Program "Education-and-Research Hub for Mathematics-for-Industry"
Program Leader

The Joint Conference of ASCM 2009 and MACIS 2009

Asian Symposium on Computer Mathematics

Mathematical Aspects of Computer and Information Sciences

COE Lecture Note Vol.22, Faculty of Mathematics, Kyushu University
ISSN 1881-4042
Editors: Masakazu Suzuki, Hoon Hong, Hirokazu Anai, Chee Yap, Yousuke Sato, Hiroshi Yoshida
Date of issue: 14 December 2009
Publisher:
Faculty of Mathematics, Kyushu University,
Global COE Program "Education-and-Research Hub for Mathematics-for-Industry"
Mathematical Research Center for Industrial Technology
Support Program for Improving Graduate School Education "Training Program of Ph.D. and Master's in Mathematics as Required by Industry"
Motooka 744, Nishi-ku, Fukuoka, 819-0395, JAPAN
Tel +81-(0)92-802-4404, Fax +81-(0)92-802-4405
URL http://gcoe-mi.jp/

Printed by
Kijima Printing, Inc.
Shirogane 2-9-6, Chuo-ku, Fukuoka, 812-0012, Japan
TEL +81-(0)92-531-7102 FAX +81-(0)92-524-4411

Foreword

Two international conferences, the 9th Asian Symposium on Computer Mathematics (ASCM 2009), and the 3rd International Conference on Mathematical Aspects of Computer and Information Sciences (MACIS 2009), are held jointly at Fukuoka on December 14th-17th 2009 supported by the GCOE program "Math-for-industry" of the Graduate School of Mathematics of Kyushu University and Mathematical Research Center for Industrial Technology (MRIT) of Kyushu University. The programs of ASCM and those of MACIS are organized independently by each program committee except for invited talks. Sessions of ASCM and those of MACIS are held in parallel and the invited talks are given in plenary sessions. There is also a satellite event associated with the conference. The Workshop on E-Inclusion in Mathematics and Science 2009 (WEIMS'09) is held at JST Innovation Plaza Fukuoka, just prior to ASCM-MACIS 2009.

The Asian Symposium on Computer Mathematics (ASCM) is a series of conferences which serve as a forum for participants to present original research, learn of research progress and developments, and exchange ideas and views on doing mathematics using computers. The previous ASCM meetings were held in Beijing, China (1995), Kobe, Japan (1996), Lanzhou, China (1998), Chiang Mai, Thailand (2000), Matsuyama, Japan (2001), Beijing, China (2003), Seoul, Korea (2005), Singapore, Singapore (2007). This year, the meeting consists of invited talks, regular sessions of contributed papers, and three organized sessions on the following topics:

1) Digitizing Mathematics − From Pen and Paper to Digital Content −
2) Validated Numerical Computation
3) Computational Algebraic Number Theory

Each organized session is run by its organizer(s) independently. Regular sessions are run in a traditional style of ASCM. Specific topics include but are not limited to:

* Computer-aided problem solving and instruction
* Symbolic, algebraic, and geometric computation
* Computational number theory, cryptography, and combinatorics
* Automated mathematical reasoning and interactive theorem proving
* Symbolic/numeric hybrid methods
* Computational algebra and geometry
* Formalization of mathematics
* Computational methods for differential and difference equations
* Mathematical software design and implementation
* Parallel/distributed/network computing
* Exact numerical methods and zero bounds
* Foundations of real computation and complexity issues

Mathematical Aspects of Computer and Information Sciences (MACIS) is a new series of conferences where foundational research on theoretical and practical problems of mathematics for computing and information processing may be presented and discussed. MACIS also addresses experimental and case studies, scientific and engineering computation, design and implementation of algorithms and software systems, and applications of mathematical methods and tools to outstanding and emerging problems in applied computer and information sciences. The first MACIS conference took place in Beijing (China), July 24-26, 2006. The second MACIS conference took place in Paris (France), December 5-7, 2007.

MACIS2009 is run in a format where each PC member organizes a session on a specific topic. MACIS 2009 consists of 3 sessions on three main themes as shown in the following:

1) Polynomial system solving (Complex/Real/Rational)
2) Systems and Control
3) Software Science

There were 39 papers submitted to ASCM regular sessions this year. The program committee selected the 26 papers appearing in these proceedings after careful evaluation including two or more referee reports per submission. Almost all the papers for the organized sessions of ASCM and MACIS were solicited by the organizers. The numbers of selected papers for organized session are as follows:

ASCM:

1) Digitizing Mathematics	6
2) Validated Numerical Computation	6
3) Computational Algebraic Number Theory	3

MACIS:

1) Polynomial system solving (Complex/Real/Rational)	8
2) Systems and Control	5
3) Software Science	6+1 (Short presentation)

We gratefully acknowledge the thorough and important work of the program committee members and referees, whose names appear on the following pages, and thank all the authors and lecturers for their contributions.

We are grateful for the support of the sponsoring organizations noted at the web page of ASCM-MACIS 2009. As for their organizational assistance we particular thank the local organizers and the COE office at Kyushu University.

December 2009

Masakazu Suzuki
Hoon Hong
Hirokazu Anai
Chee Yap
Yosuke Sato
Hiroshi Yoshida
Tatsuyoshi Hamada
Koji Nakagawa
Kazuhiro Yokoyama

Conference Organization

General Chair

Masakazu Suzuki	Kyushu University, Japan

Program Committee Chairs of the ASCM

Chee Yap	New York University, USA
Yosuke Sato	Tokyo University of Science, Japan

Program Committee Chairs of the MACIS

Hoon Hong	North Carolina State University, USA
Hirokazu Anai	Kyushu University/Fujitsu Laboratories LTD, Japan

Coordinator of the joint conference

Kazuhiro Yokoyama	Rikkyo University, Japan

Program Committee Members of the ASCM

Xavier Dahan	Kyushu University, Japan
Xiao-Shan Gao	Chinese Academy of Sciences, China
Deepak Kapur	Univ. of New Mexico, USA
Ziming Li	Chinese Academy of Sciences, China
Hirokazu Murao	The University of Electro Communications, Japan
Mitsuhiro Nakao	Kyushu University, Japan
Hyungju Park	KIAS, Korea
Guenael Renault	UPMC, INRIA France
Ko Sakai	University of Tsukuba, Japan
Alan P. Sexton	University of Birmingham, UK
Volker Sorge	University of Birmingham, UK
Gert Vegter	Groningen University, the Netherlands

Program Committee Members of the MACIS

Enric Rodriguez Carbonell	Technical University of Catalonia, Spain
Andrzej Cichocki	RIKEN, Japan
Mohab Safey El Din	UPMC, INRIA, France
Hiroshi Yoshida	Kyushu University, Japan
Jeremy Johnson	Drexel University, USA
Masaaki Kanno	Niigata University, Japan
Gabriel Dos Reis	Texas A & M University, USA

Fabrice Rouillier	INRIA, France
Eric Schost	Univ. of Western Ontario, Canada

External Reviewers

Michael Kerber	Guillaume Moroz
Paul Vrbik	Katsusuke Nabeshima
Janwa Heeralal	Takashi Kako
Kinji Kimura	Kosaku Nagasaka
Takeshi Osoekawa	Adrien Poteaux
Kiyoshi Shirayanagi	Akira Suzuki
Gilles Villard	Dingkang Wang
Chun-Ming Yuan	Lihong Zhi

Invited Speakers

Markus Rosenkranz	Johann Radon Institute for Computational and Applied Mathematics, Austria (Joint work with Professor Bruno Buchberger)
Toshinori Oaku	Tokyo Woman's Christian University, Japan
Kokichi Sugihara	Meiji University, Japan
Lihong Zhi	Academy of Mathematics and System Sciences, China

Local chairs

Hiroshi Yoshida	Kyushu University, Japan
Tatsuyoshi Hamada	Fukuoka University/JST CREST, Japan
Koji Nakagawa	Kyushu University, Japan

Sponsors

Global COE Program, Mathematical Research Centre for Industrial Technology, Kyushu University

SPARC Japan

CYBERNET

Japan Society of Symbolic and Algebraic Computations

MapleSoft

Microsoft

JAL HOTELS

Table of Contents

Invited talks

ASCM 2009 Regular Session

ASCM Organized Session

Digitizing Mathematics

Validated Numerical Computation

Computational Algebraic Number Theory

MACIS2009

Polynomial system solving

Systems and Control

Software Science

Invited talks

A New Symbolic Method for Linear Boundary Value Problems Using Groebner Bases

Markus Rosenkranz[1]
(joint work with Bruno Buchberger[2],
Georg Regensburger[1], and Loredana Tec[2])

[1] Johann Radon Institute for Computational and Applied Mathematics,
Austrian Academy of Sciences, Altenberger Str. 69, 4040 Linz, Austria, Europe
Markus.Rosenkranz@oeaw.ac.at, Georg.Regensburger@oeaw.ac.at

[2] Research Institute for Symbolic Computation,
Johannes Kepler University, 4032 Castle of Hagenberg, Austria, Europe
Bruno.Buchberger@risc.uni-linz.ac.at, ltec@risc.uni-linz.ac.at

Abstract

Boundary value problems are of utmost importance for science and engineering. In fact, most differential equations come along with boundary conditions of some sort. It is therefore surprising that such problems—even in the linear case—have gained little attention in Symbolic Computation. Consequently, their coverage in computer algebra systems is rather unsystematic and unpredictable.

The proper consideration of boundary conditions leads to a substantial revision of the algebraic structures currently used in established symbolic methods like differential algebra or differential Galois theory. One important ingredient in an algebraic approach to boundary value problems is the interaction of differential, integral and boundary operators. We present one such approach, based on Buchberger's powerful concept of Groebner bases.

For the implementation of the method we use the functor concept introduced by Buchberger for the Theorema system. This allows for easy adjustment of the code to various coefficient domains and different representations of the underlying objects.

Holonomic functions revisited

Toshinori Oaku

Tokyo Woman's Christian University
2-6-1, Zempukuji, Suginami-ku, Tokyo, 167-8585 Japan
oaku@lab.twcu.ac.jp

Abstract

A holonomic function is a differentiable or generalized function which satisfies a holonomic system of linear partial or ordinary differential equations with polynomial coefficients. We present algorithms for computing systems of differential equations which the sum, the product, the restriction, and the integration of holonomic functions satisfy. These algorithms are based on Gröbner bases of differential operators and are rigorous in the sense that the output differential equtaions are also holonomic.

1 Differential operators and holonomic systems

Let us denote by D_n the ring of differential operators on the variables $x = (x_1, \dots, x_n)$ with polynomial coefficients. An element P of D_n is written in a finite sum

$$P = \sum_{\alpha \in \mathbb{N}^n} a_{\alpha,\beta} x^\alpha \partial^\beta, \tag{1}$$

where $\alpha = (\alpha_1, \dots, \alpha_n)$, $\beta = (\beta_1, \dots, \beta_n) \in \mathbb{N}^n$ are vectors of nonnegative integers with $\mathbb{N} = \{0, 1, 2, \dots\}$, $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, $\partial^\beta = \partial_1^{\beta_1} \cdots \partial_n^{\beta_n}$ with the derivations $\partial_i = \partial/\partial x_i$ $(i = 1, \dots, n)$, and $a_{\alpha,\beta}$ are complex numbers.

Given $P_1, \dots, P_r \in D_n$, we associate the left ideal $I = D_n P_1 + \cdots + D_n P_r$ generated by $P_1, \dots, P_r$ with a system of linear differential equations

$$P_1 u = \cdots = P_r u = 0 \tag{2}$$

for an unknown function u. This enables us to work with a left ideal of D_n instead of each system of linear differential equations. Here we suppose that the unknown function u belongs to a 'function space' $\mathcal{F}$ which is a left D_n-module. Examples of such $\mathcal{F}$ are the set $C^\infty(U)$ of C^∞ functions on an open subset U of $\mathbb{R}^n$, the set $\tilde{\mathcal{O}}(U)$ of possibly multi-valued analytic functions on an open subset U of $\mathbb{C}^n$, the set $\mathcal{D}'(U)$ of the Schwartz distributions on an open subset U of $\mathbb{R}^n$, and the set $S'(\mathbb{R}^n)$ of tempered distributions.

A weight vector for D_n is an integer vector

$$w = (w_1, \dots, w_n; w_{n+1}, \cdots w_{2n}) \in \mathbb{Z}^{2n}$$

with the conditions $w_i + w_{n+i} \geq 0$ for $i = 1, \dots, n$, which are necessary in view of the commutation relation $\partial_i x_i = x_i \partial_i + 1$ in D_n. For a nonzero differential operator P of the form (1), we define its *w-order* to be

$$\mathrm{ord}_w(P) := \max\{\langle w, (\alpha, \beta)\rangle = w_1\alpha_1 + \cdots + w_n\alpha_n + w_{n+1}\beta_1 + \cdots + w_{2n}\beta_n \mid a_{\alpha,\beta} \neq 0\},$$

and its *w-initial part* to be

$$\mathrm{in}_w(P) := \sum_{\langle w, (\alpha,\beta)\rangle = \mathrm{ord}_w(P)} a_{\alpha,\beta} x^\alpha \partial^\beta.$$

In particular, when $w = (\mathbf{0}, \mathbf{1}) = (0, \ldots, 0; 1, \ldots, 1)$, then the polynomial

$$\sigma(P)(x,\xi) := \sum_{\langle w,\,(\alpha,\beta)\rangle = \mathrm{ord}_w(P)} a_{\alpha,\beta} x^\alpha \xi^\beta \ \in \mathbb{C}[x,\xi]$$

is called the *principal symbol* of P.

Definition 1 A left ideal I of D_n is said to be *holonomic* if the ideal $\sigma(I)$ of $\mathbb{C}[x,\xi]$ which are generated by the set $\{\sigma(P)(x,\xi) \mid P \in I,\, P \neq 0\}$ is of dimension n, that is, the characteristic variety of I, which is defined to be

$$\mathrm{Char}(I) := \{(x,\xi) \in \mathbb{C}^{2n} \mid p(x,\xi) = 0 \text{ for any } p \in \sigma(I)\},$$

is of dimension n. (In general, the dimension of $\mathrm{Char}(I)$ is greater than or equal to n if $I \neq D_n$.) We call (2) a *holonomic system* if the left ideal $I = D_nP_1 + \cdots + D_nP_n$ is holonomic. We also call $\mathrm{Char}(I)$ the characteristic variety of the holonomic system (2).

The characteristic variety of (2) can be computed by Gröbner bases ([4]). The dimension of the characteristic variety can be computed by using the Hilbert function.

2 Holonomic functions

Definition 2 Let u be a C^∞ function or a distribution (in the sense of L. Schwartz, or a generalized function in the sense of Gelfand-Shilov) defined on an open subset U of $\mathbb{R}^n$. Then we call u a *holonomic function* or a *holonomic distribution* on U if u satisfies a holonomic system. In other words, u is holonomic if and only if its *annihilator*

$$\mathrm{Ann}_{D_n} u := \{P \in D_n \mid Pu = 0 \text{ on } U\}$$

is a holonomic ideal.

For example, given an arbitrary polynomial f, the C^∞ function e^f is holonomic on $\mathbb{R}^n$. In fact we can easily verify that

$$\mathrm{Ann}_{D_n} e^f = D_n\left(\partial_1 - \frac{\partial f}{\partial x_1}\right) + \cdots + D_n\left(\partial_n - \frac{\partial f}{\partial x_n}\right),$$

which shows that the characteristic variey is $\{(x,\xi) \in \mathbb{C}^{2n} \mid \xi_1 = \cdots = \xi_n = 0\}$.

If $f_1, \ldots, f_m$ are nonzero polynomials with real coefficients, then the distribution $u = (f_1)_+^{\lambda_1} \cdots (f_m)_+^{\lambda_m}$ defined by

$$\langle u, \varphi\rangle = \int_{f_1 \geq 0, \ldots, f_m \geq 0} f_1(x)^{\lambda_1} \cdots f_m(x)^{\lambda_m} \varphi(x)\, dx \quad (\varphi \in C_0^\infty(\mathbb{R}^n))$$

is holonomic on $\mathbb{R}^n$ unless $(\lambda_1, \ldots, \lambda_m) \in \mathbb{C}^m$ is not contained in an exceptional set where u cannot be defined. There is an algorithm to compute a holonomic ideal of which u is a solution if the coefficients of f_i are contained in a computable field. In particular, the product of Heaviside's functions

$$Y(f_1) \cdots Y(f_m) = (f_1)_+^0 \cdots (f_m)_+^0$$

is a holonomic distribution on $\mathbb{R}^n$. Elementary functions are not necessarily holonomic. For example, the smooth (C^∞) function x^y in the two variables x and y defined on $\{(x,y) \in \mathbb{R}^2 \mid x > 0,\ y \in \mathbb{R}\}$ is not holonomic in the sense above.

3 Operations on holonomic functions

For the sake of simplicity, let us assume that u and v are holonomic functions or distributions defined on the whole $\mathbb{R}^n$. Then the following functions or distributions are holonomic under the condition that they are well-defined. Moreover, if a holonomic ideal for u (and a one for v if relevant) is explicitly given, then there exist algorithms to compute a holonomic ideal which each of the following functions satisfies.

(1) Pu with $P \in D_n$.

(2) The sum $u + v$.

(3) The restriction $u|_Y$ of u to an affine subspace of $\mathbb{R}^n$ if it is well-defined as in the case that u is smooth.

(4) The product uv if it is well-defined as in the case that u is C^∞ and v is a distribution.

(5) The definite integral $\displaystyle\int_{\mathbb{R}^{n-d}} u(x_1, \ldots, x_{n-d}, x_{n-d+1}, \ldots, x_n)\, dx_{n-d+1} \cdots dx_n$ with parameters if it is well-defined as in the case that u has a compact support with respect to the integration variables.

Let us explain briefly how to compute holonomic ideals for functions above. Let I and J be holonomic ideals for u and v respectively. First, a holonomic ideal for Pu can be computed as an ideal quotient $I : P$, and a one for $u+v$ as an ideal intersection $I \cap J$, both by using Gröebner bases.

The restriction algorithm was given in [5] for one codimensional case and in [8] for the general case. For this, one needs a Gröbner base of the ideal I with respect to an ordering compatible with the weight vector of type

$$w = (0, \ldots, 0, -1, \ldots, -1; 0, \ldots, 0, 1, \ldots, 1).$$

A holonomic ideal for $u(x)v(y)$ is generated by I and J over D_{2n}. Then by restricting $u(x)v(y)$ to the diagonal $x = y$, we obtain a holonomic ideal. Finally, the definite integral with parameters can be computed also by the restriction algorithm applied to the Fourier transform of I (cf. [7], [9], [6]).

The theory of D-modules assures that the functions obtained by the operations (1)–(5) above are holonomic (See for example, [2]). The outputs of our algorithms are also holonomic since they follow the D-module theoretic procedures precisely, not in a heuristic way as in the pioneering works of Almkvist-Zeilberger [1] and Takayama [10].

4 Definite integrals with the Heaviside function

Let u be a holonomic function on $\mathbb{R}^n$. Let $f_1, \ldots, f_m$ be nonzero polynomials in $x = (x_1, \ldots, x_n)$ with real coefficients. Setting

$$D(x_1, \ldots, x_d) = \{(x_{d+1}, \ldots, x_n) \in \mathbb{R}^{n-d} \mid f_1(x) \geq 0, \ldots, f_m(x) \geq 0\},$$

let us consider the definite integral

$$v(x_1, \ldots, x_d) = \int_{D(x_1, \ldots, x_d)} u(x)\, dx = \int_{\mathbb{R}^n} Y(f_1) \cdots Y(f_m) u(x)\, dx.$$

We suppose that this integral is well-defined. This is the case, for example, when u is smooth and $D(x_1, \ldots, x_d)$ is compact for any $(x_1, \ldots, x_d) \in \mathbb{R}^d$. Then a holonomic ideal for this integral can be computed by combining the algorithms explained so far. If the integrand is the exponential of a polynomial, a power of a polynomial, or the product of such functions, then there are some shortcuts to follow. For practical computations, a library file `nk_restriction.rr` of a computer algebra system Risa/Asir provides one of the most efficient implementations of the restriction and integration algorithms.

Example 3 Consider the definite integral

$$v(t) = \int_{D(t)} \sqrt{x+t}\, dxdy = \int_{\mathbb{R}^2} Y(1-x^2-y^2)(x+t)_+^{\frac{1}{2}}\, dxdy$$

with $D(t) = \{(x, y) \in \mathbb{R}^2 \mid x^2+y^2 \leq 1,\ x+t \geq 0\}$. Then by the algorithm described above, we know that $v(t)$ satisfies a linear ordinary differential equation

$$4(1-t^2)\frac{d^2v}{dt^2} + 8t\frac{dv}{dt} - 5v = 0 \tag{3}$$

as a distribution on $\mathbb{R}$. In fact, $v(t)$ is continuous on $\mathbb{R}$ but may fail to be infinitely differentiable at the singular points $t = \pm 1$ of the equation (3).

References and Notes

[1] Almkvist, G., Zeilberger, D., The method of differentiating under the integral sign. J. Symbolic Computation **10** (1990), 571–591.

[2] Björk, J.-E., Rings of Differential Operators. North-Holland, 1979.

[3] Nakayama, H., `nk_restriction.rr`: a library file of Risa/Asir, `http://www.math.kobe-u.ac.jp/Asir/asir.html` (2009)

[4] Oaku, T., Computation of the characteristic variety and the singular locus of a system of differential equations with polynomial coefficients. Japan J. Indust. Appl. Math. **11** (1994), 485–497.

[5] Oaku, T., Algorithms for b-functions, restrictions, and algebraic local cohomology groups of D-modules. Advances in Appl. Math. **19** (1997), 61–105.

[6] Oaku, T., Shiraki, Y., Takayama, N., Algebraic algorithms for D-modules and numerical analysis. Computer mathematics (Proceedings of ASCM 2003), 23–39, Lecture Notes Ser. Comput., 10, World Sci. Publ., River Edge, NJ, 2003.

[7] Oaku, T., Takayama, N., An algorithm for de Rham cohomology groups of the complement of an affine variety. J. Pure Appl. Algebra **139** (1999), 201–233.

[8] Oaku, T., Takayama, N., Algorithms for D-modules — restriction, tensor product, localization, and local cohomology groups. J. Pure Appl. Algebra **156** (2001), 267–308.

[9] Saito, M., Sturmfels, B., Takayama, N., Gröbner Deformations of Hypergeometric Differential Equations. Springer Verlag, 2000.

[10] Takayama, N., An approach to the zero recognition problem by Buchberger algorithm. J. Symbolic Computation **14** (1992), 265–282.

Computational Illusion
Toward Escher and Beyond Escher

Kokichi Sugihara

Meiji Institute for Advanced Study of Mathematical Sciences, Meiji University,
1-1-1 Higashimita, Tama-ku, Kawasaki, Kanagawa 214-8571, Japan
kokichis@isc.meiji.ac.jp

.....

Abstract

M. C. Escher, a Dutch woodcut artist, is one of a few artists who utilized mathematical structures in creating artworks. We study three groups of Escher's works, isohedral-tiling art, Sky-and-Water-type art, and pictures of impossible objects, from a mathematical point of view. In particular, we consider how to generate Escher-like art by purely mathematical algorithms, and how to extend them to generalize his art patterns.
Keywords: Illusion, Escher, tiling, impossible objects, figure–ground reversal.

1 Introduction

Mathematics and art are usually very far apart, but come close together in some special cases. M. C. Escher, a Dutch woodcut artist, is a remarkable artist who shortened the distance between mathematics and art by explicitly utilizing mathematical structures. Typical examples are isohedral tiling in "Regular Division of the Plane No. 56 (Lizard)" (1942), hyperbolic-space tiling in "Circle Limit IV" (1960), continuously changing tiling in "Sky and Water I" (1938) and unrealizable motion in "Waterfall" (1961).

Indeed, Escher's artworks have been analyzed from a mathematical point of view in a variety of ways. Escher's tiling artworks were classified according to the mathematical categorization of isohedral tilings [3, 7]. Creation of Escher-like tiling patterns by computers has also been tried for isohedral tiling by Cervini et al. [1] and Kaplan and Salesin [4, 5].

However, simple introduction of mathematical structures to art might just generate abstract patterns. What is remarkable in Escher is that he combined mathematical structures with optical illusions, and thus made mathematical structures nontrivial from an artistic point of view. The visual effects used by Escher include continuous morphing, animal-like complicated tiles, figure–ground reversal, and impossible objects and motions.

Therefore, to generate Escher-like art using computers, we must consider not only geometric structures but also the effects of visual illusion.

We show three examples of trials to combine mathematical structures and visual illusion aiming at computer-aided systems to generate Escher-like art patterns. They are design of complicated isohedral tilings, generation of Sky-and-Water-type tilings, and three-dimensionalization of impossible objects and impossible motions, which are presented in Sections 2, 3 and 4, respectively.

2 Isohedral Tiling and Faithful Escherization

The first group of Escher's works we consider is tiling art with mutually congruent tiles. This group is based on a geometric structure called isohedral tilings.

A partition of the plane into topological disks and their boundaries is called a *tiling*. A tiling is said to be *monohedral* if all the tiles are congruent. A monohedral tiling is said to be *isohedral* if there is a subgroup of the group of congruent transformations in the plane such that the tiling is invariant under those transformations and any pair of tiles has a congruent transformation in the subgroup that maps one to the other.

In isohedral tiling, the relation of one tile with the surrounding tiles around it is the same for every tile. Hence, the same deformation rules can be applied to all the tiles simultaneously and the resulting structures remain a tiling. This way, we can modify the shape of the tiles into a complicated one such as an animal or a human. This is what Escher did in his work notes. In other words, Escher started with a simple initial tiling, and modified the tiles to the shape he wanted.

In computers, on the other hand, we can move in the opposite direction. That is, we first choose an arbitrary goal shape, and next search for a tile that is similar to the goal shape and that admits an isohedral tiling. In this process, visually interesting tilings will be generated if we can find tiles that are close to the goal shape.

Suppose that we are given a figure represented by a cyclic sequence of n points on the boundary of the figure. Let W be a $2 \times n$ matrix composed of the x and y coordinates of the n points. We call W the *goal shape*. Let U be another $2 \times n$ matrix such that:

$$U = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{pmatrix}, \tag{1}$$

by which we want to represent the shape of a tile that admits an isohedral tiling and that is close to W. That is, U is the tile we want to search for; so the entities of U are known variables. We assume that both shapes are placed so that their center of gravity coincides with the origin of the coordinate system.

To measure the distance between U and W, we define the index:

$$D^2(U, W) = \min_{s,\theta} \left\| sR(\theta)\frac{U}{\|U\|} - \frac{W}{\|W\|} \right\|^2, \tag{2}$$

where $\|X\|$ denotes the Frobenius norm of matrix X, s is a scalar representing the scale transformation, and $R(\theta)$ represents the rotation of the figure by angle θ around the origin.

The above distance is convenient because we can rewrite it as:

$$D^2(U, W) = 1 - \frac{\|UW^{\mathrm{T}}\|^2 + 2\det(UW^{\mathrm{T}})}{\|U\|^2\|W\|^2}, \tag{3}$$

and thus we can remove the minimum operation with respect to s and θ [11].

We want to find a shape U that admits an isohedral tiling. Isohedral tilings were classified into 93 types and they are represented by symbols IH01, IH02, ..., IH93 [2]. If we fix the type of the isohedral tiling, we can represent the constraint as a relationship among the points on the boundary of U. For example, suppose that U admits the isohedral tiling of type IH07, which is generated by rotations by $2\pi/3$ around two points. Let p_i, p_j, p_k be three points on the boundary of U such that rotating the plane around p_j by $2\pi/3$ results in p_i moving to the position previously occupied by p_k. Then:

$$R(2\pi/3)(p_i - p_j) = p_k - p_j, \tag{4}$$

which is represented by linear constraints with respect to the x and y coordinates of the three points x_i, y_i, x_j, y_j, x_k and y_k. Collecting similar constraints:

$$A\boldsymbol{u} = 0, \tag{5}$$

where $\boldsymbol{u} = (x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)^{\mathrm{T}}$ and A is a constant matrix associated with the isohedral tiling type IH07. For other isohedral tilings, we similarly obtain linear constraints depending on the type.

On the other hand, the second term of eq. (3) can be rewritten as:

$$\frac{\|UW^{\mathrm{T}}\|^2 + 2\det(UW^{\mathrm{T}})}{\|U\|^2\|W\|^2} = \frac{\boldsymbol{u}^{\mathrm{T}} V \boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}, \tag{6}$$

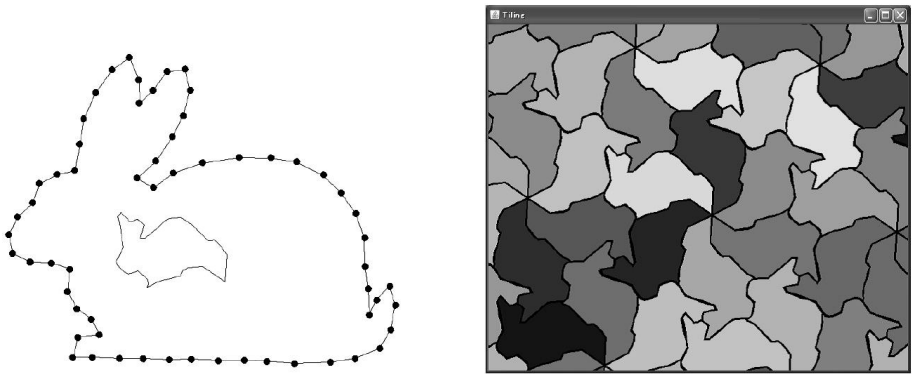
where V is a symmetric matrix depending on W.

Therefore, if we fix the type of the isohedral tiling, the problem of finding the tile that is similar to the goal shape and that admits isohedral tiling can be reduced to the optimization problem:

$$\begin{array}{ll} \text{maximize} & \dfrac{\boldsymbol{u}^{\mathrm{T}} V \boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}} \\ \text{subject to} & A\boldsymbol{u} = 0, \end{array}$$

which can be solved efficiently [6].

Figure 1 shows an example of a tiling obtained by this method: (a) shows a pair of a goal shape (the larger shape) and the tile found by our method (the smaller shape), and (b) shows the resulting tiling.



(a) (b)

Figure 1: Isohedral tiling generated from a goal shape of a rabbit.

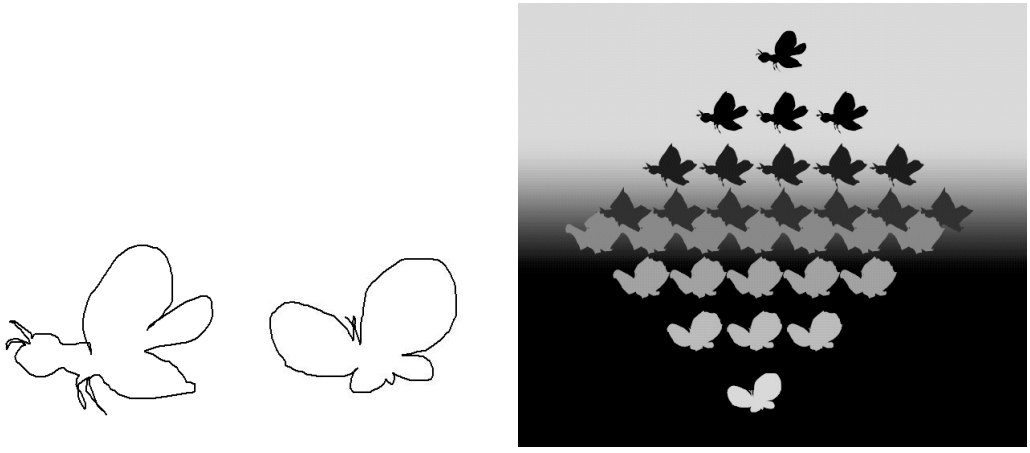
3 Morphing for Sky-and-Water Tilings

The second group of Escher's works we consider is the Sky-and-Water- type tiling, in which a figure at the top changes its shape gradually downward and melts away into the background, and another shape gradually appears from the background. This group of artworks is a combination of tiling, morphing, and figure–ground reversal.

Escher constructed this type of tiling by first generating a dihedral tiling at the intermediate level, and deforming it so that one tile gradually becomes a clear object upward, and the other tile gradually becomes the other clear object downward [8].

In computers, on the other hand, we first fix two goal shapes at the top and at the bottom, and generate intermediate tiles to form the Sky-and-Water tiling pattern. For this purpose, the morphing is applied to the top figure and to the shape of the vacant space surrounded by four copies of the bottom figure [10].

Figure 2 shows an example of the tiling generated in this way: (a) shows a pair of top and bottom figures, and (b) shows the computer-generated tiling.



(a) (b)

Figure 2: Sky-and-Water tiling generated from a bee and a butterfly.

4 Realization of Impossible Objects and Impossible Motions

The third group of Escher's works is the class of pictures using impossible objects. A typical example is "Waterfall" (1961), in which water that falls down runs through a water path upward to the start point of the fall again, and thus the water runs in a cyclic manner forever. This is physically impossible because it contradicts the impossibility of an eternal engine.

Escher drew his impossible objects and impossible motions only on a plane. On the other hand, we found that some of those pictures are realizable as actual objects and actual motions in the three-dimensional world through optical illusion.

The picture lacks depth information, and hence there is great freedom in reconstructing the three-dimensional objects from the picture. Therefore, even if a picture looks impossible, we can sometimes construct a solid, although the resulting solid is quite different from our intuition. With the same trick, we can also create impossible-looking motions. Such impossible objects and motions can be searched for by a computer in the following manner.

Suppose that the viewpoint is fixed at the origin of the xyz Cartesian coordinate system, and the picture is fixed on the plane $z = 1$. Let $v_i = (x_i, y_i, 1)$ be the ith vertex in the picture, and let $(x_i/t_i, y_i/t_i, 1/t_i)$ be the associated vertex on the three-dimensional solid whose projection coincides with the given picture, where t_i is an unknown variable

representing the depth of v_i. Furthermore, let the jth face of the solid be represented by the equation:

$$a_j x + b_j y + c_j z + 1 = 0. \tag{7}$$

Let m be the number of vertices and n be the number of faces represented in the picture.

Suppose that the ith vertex is on the jth face. Then:

$$a_j x_i + b_j y_i + c_j + t_i = 0. \tag{8}$$

For all pairs of vertices and faces containing them, we obtain similar equations, and thus we can construct a system of linear equations, which we represent by:

$$A\boldsymbol{w} = 0, \tag{9}$$

where $\boldsymbol{w}$ is a vector of unknown parameters:

$$\boldsymbol{w} = (t_1, t_2, \ldots, t_m, a_1, b_1, c_1, \ldots, a_n, b_n, c_n)^{\mathrm{T}}, \tag{10}$$

and A is a constant matrix.

Next, suppose that the kth vertex is behind the lth face when it is extended. Then:

$$a_j x_i + b_j y_i + c_j + t_i < 0. \tag{11}$$

Collecting all such inequalities yields a system of linear inequalities:

$$B\boldsymbol{w} > 0. \tag{12}$$

The set of all solids that generate a given projected picture is represented by the feasible region specified by the linear equations (9) and inequalities (12).

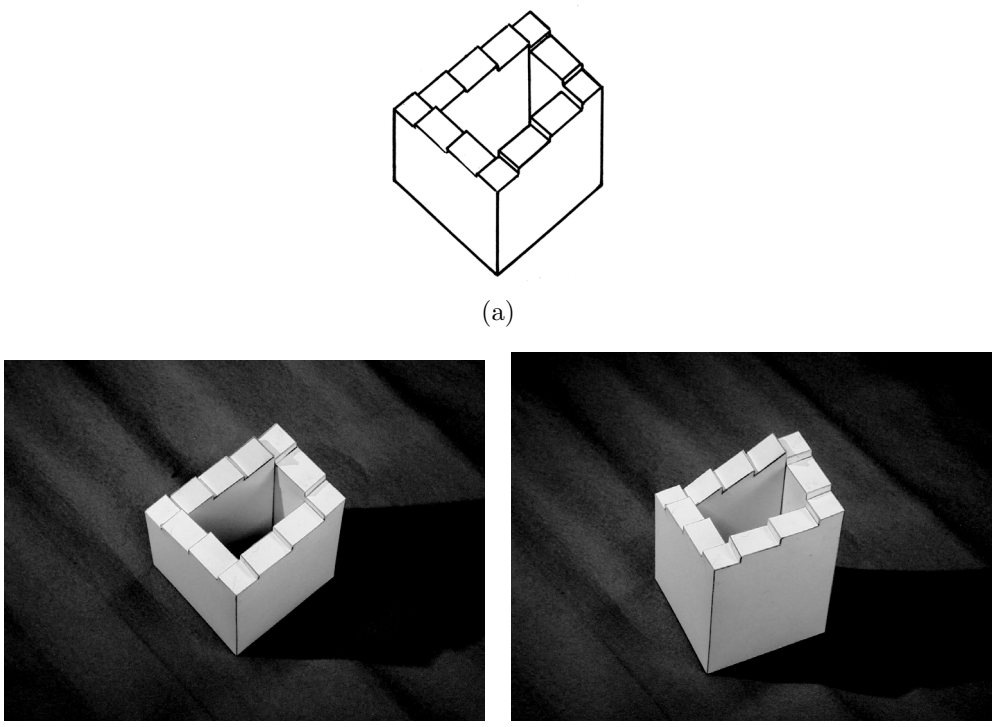
This feasible region is fragile in the sense that it may become empty when even a slight numerical error occurs in the plane. However, we constructed a method to snap the vertex positions in the picture to the correct locations, and thus established a robust method to judge the realizability of solids from a picture [9]. Using this method, we can construct solids for some pictures of "impossible objects" and can also generate impossible motions using those solids.

Figure 3 shows an example of a realization of an impossible object: (a) shows a picture of an impossible object, (b) shows a realization of the associated solid, and (c) shows the same solid seen from another angle.

5 Concluding Remarks

We have presented three examples of computer-aided methods, by which we can generate Escher-like tilings and three-dimensional solids that realize Escher-like impossible objects and motions. The success of these computer-aided methods means that not only can we realize the mathematical structures behind Escher's works, we can also realize visual effects by applying computer power to the search for the optimal shapes.

Escher also generated many other types of artwork, most of which are also based on mathematical structures. Hence, there still remain many possibilities for utilizing computer power to generate those artistic patterns automatically and effectively.



(a) (b) (c)

Figure 3: Realization of a solid that appears to be an impossible object.

Acknowledgments

The computer-aided isohedral tiling presented in Section 2 is a joint work with Mr. H. Koizumi, a Master's student of the Department of Mathematical Informatics of the University of Tokyo. This work is supported by the Grant-in-Aid for Scientific Research (B) No. 20360044 and the Grant-in-Aid for Exploratory Research No. 19650003 of the Japanese Society for Promotion of Science.

References and Notes

[1] L. Cervini, R. Farinato and L. Loreto: The interactive computer graphics (ICG) production of the 17 two-dimensional crystallographic groups, and other related topics. In H. S. M. Coxeter, M. Emmer, R. Penrose and M. L. Teuber (eds.): *M. C. Escher — Art and Science*, pp. 269–284, North-Holland, Amsterdam, 1986.

[2] B. Grünbaum and G. C. Shephard: *Tiling and Patterns — An Introduction*. W. H. Freeman, New York, 1989.

[3] C. S. Kaplan: Metamorphosis in Escher's art. In *Bridges 2008 — Mathematical Connections in Art, Music and Science*, July 24-28, Leeuwarden, the Netherlands, pp. 39–46, 2008.

[4] C. S. Kaplan and D. H. Salesin: Escherization. *Proceedings of SIGGRAPH 2000*, New Orleans, LA, pp. 499–510, 2000.

[5] C. S. Kaplan and D. H. Salesin: Dihedral Escherization. *Proceedings of Graphics Interface 2004*, May 2004, London, Ontario, pp. 255–262, 2004.

[6] H. Koizumi and K. Sugihara: Computer aided design system for Escher-like tilings. *Proceedings of the 25th European Workshop on Computational Geometry*, March 2009, Brussels, pp. 345–348, 2009.

[7] D. Schattschneider: *M. C. Escher — Visions and Symmetry*, New Edition. Abrams, New York, 2004.

[8] D. Schattschneider: Lessons in duality and symmetry from M. C. Escher. In *Bridges 2008 — Mathematical Connections in Art, Music and Science*, July 24-28, Leeuwarden, the Netherlands, pp. 1–8, 2008.

[9] K. Sugihara: Computer-aided creation of impossible objects and impossible motions. *The Kyoto International Conference on Computational Geometry and Graph Theory*, Kyoto, June 2007.

[10] K. Sugihara: Computer-aided generation of Escher-like Sky and Water tiling patterns. *Journal of Mathematics and the Art* (to appear).

[11] M. Werman and D. Weinshall: Similarity and affine invariant distances between 2D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, pp. 810–814, 1995.

A Symbolic-numeric Algorithm for Computing the Multiple Roots of Polynomial Systems Accurately

Lihong Zhi[1]

[1] Key Lab of Mathematics Mechanization, AMSS,
Beijing 100190, China
lzhi@mmrc.iss.ac.cn

Extended Abstract

Consider an ideal I generated by a polynomial system $F = \{f_1, \ldots, f_t\}$, where $f_i \in \mathbb{C}[x_1, \ldots, x_s]$, $i = 1, \ldots, t$. For a given isolated singular solution $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_s)$ of F, suppose Q is the isolated primary component whose associate prime is $P = (x_1 - \hat{x}_1, \ldots, x_s - \hat{x}_s)$, in [Wu and Zhi, 2008a], we use symbolic-numeric method based on the geometric jet theory of partial differential equations introduced in [Bonasia et al., 2004, Reid et al., 2003, Zhi and Reid, 2004] to compute the index ρ and multiplicity μ, such that $Q = (I, P^\rho)$ and $\mu = \dim(\mathbb{C}[\mathbf{x}]/Q)$. We also derive a simple involutive criterion based on the special structure of the ideal (I, P^k) and apply it to the truncated coefficient matrices formulated from the Taylor series expansions of polynomials in prolonged systems of F at $\hat{\mathbf{x}}$ to order k. The number of columns of these coefficient matrices is fixed by $\binom{k+s-1}{s}$. A basis for the Max Noether space of I at $\hat{\mathbf{x}}$ is obtained from the null space of the truncated coefficient matrix of the involutive system.

If a singular solution is only known with limited accuracy, by choosing a tolerance, we can compute the index, multiplicity and a basis of the Max Noether space for this approximate singular solution. It is well known that numeric computations deeply depend on the choice of tolerance. In order to obtain accurate information about the multiplicity structure, we present in [Wu and Zhi, 2008a] an algorithm to improve the accuracy of the singular root. Suppose $\hat{\mathbf{x}} = \hat{\mathbf{x}}_{\text{exact}} + \hat{\mathbf{x}}_{\text{error}}$, where $\hat{\mathbf{x}}_{\text{exact}}$ denotes the exact singular solution of F and $\hat{\mathbf{x}}_{\text{error}}$ denotes the error in the solution, we compute the truncated coefficient matrix of the involutive system by shifting the coefficient matrix formulated from the truncated multivariate Taylor series expansions of polynomials in F at $\hat{\mathbf{x}}$ to order ρ, then generate multiplication matrices from its null vectors. Let $\hat{\mathbf{y}}$ be the averages of the traces of the multiplication matrices. In [Wu and Zhi, 2008b], we prove that if the given singular solution satisfies $\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_{\text{exact}}\| = \mathcal{O}(\varepsilon)$, and the index and multiplicity of the singular solution are computed correctly, then the refined solution obtained by adding $\hat{\mathbf{y}}$ to $\hat{\mathbf{x}}$ will satisfy $\|\hat{\mathbf{x}} + \hat{\mathbf{y}} - \hat{\mathbf{x}}_{\text{exact}}\| = \mathcal{O}(\varepsilon^2)$. If we underestimate or overestimate the index due to poorly chosen tolerance, then we can rediscover the correct index after the accuracy of the approximate singular solution improved after one or two iterations.

The size of these coefficient matrices in [Wu and Zhi, 2008a,b] is bounded by $t\binom{\rho+s-1}{s} \times \binom{\rho+s-1}{s}$ which will be very big when ρ or s is large. In general $\rho \leq \mu$, however, when the corank of the Jacobian matrix $J(\hat{\mathbf{x}})$ is one, then $\rho = \mu$, which is also called the breadth one case in [Dayton et al., 2009, Dayton and Zeng, 2005], the sizes of the matrices can easily exceed the storage capacity of a personal computer. This is the main motivation for us to consider whether we can compute the multiplicity structure of $\hat{\mathbf{x}}$ efficiently in this worst

*This is the joint work with my Ph.D students Xiaoli Wu and Nan Li.

†Supported by NKBRPC (2004CB318000) and the Chinese National Natural Science Foundation under Grant 10401035 and 60821002/F02.

case. In Li and Zhi [2009], we prove in the breadth one case, the number of free parameters used in computing each order of the Max Noether condition of I at $\hat{\mathbf{x}}$ can be reduced to $s-1$. Therefore, in order to determine a closed basis of the Max Noether space incrementally for k from 2 to $\mu-1$, we only need to check whether a computed vector $\mathbf{p}_k$ can be written as a linear combination of the last $s-1$ linear independent columns of the Jacobian matrix $J(\hat{\mathbf{x}})$. If $\mathbf{p}_k$ is not consistent with $J(\hat{\mathbf{x}})$, then we are finished. Otherwise, the coefficients of the linear combination will give us unique values of those free parameters and we find a new Max Noether condition of the k-th order. The size of matrices we used in computing each order of the Max Noether conditions is bounded by $t \times (s-1)$. It does not depend on the multiplicity. Moreover, during the computation, we only need to store polynomials, the LU decomposition of the last $s-1$ columns of the Jacobian matrix and the computed Max Noether conditions. Therefore, in the breadth one case, both storage space and execution time for computing a closed basis of the Max Noether space are reduced significantly. We also apply these strategies to reduce the matrices appeared in [Wu and Zhi, 2008a] to obtain a more efficient algorithm for refining an approximately known multiple root for this special case.

References and Notes

Bonasia, J., Lemaire, F., Reid, G., Scott, R., and Zhi, L. Determination of approximate symmetries of differential equations. In *CRM Proceedings and Lecture Notes*, pages 233–249, 2004.

Dayton, B., Li, T., and Zeng, Z. Multiple zeros of nonlinear systems. `http://orion.neiu.edu/ zzeng/Papers/MultipleZeros.pdf`, 2009.

Dayton, B. and Zeng, Z. Computing the multiplicity structure in solving polynomial systems. In *ISSAC'05 Proc. 2005 Internat. Symp. Symbolic Algebraic Comput.*, pages 116–123, 2005.

Li, N. and Zhi, L. Compute the multiplicity structure of an isolated singular solution: Case of breadth one. Manuscript, 2009.

Reid, G., Tang, J., and Zhi, L. A complete symbolic-numeric linear method for camera pose determination. In Sendra, J., editor, *Proc. 2003 Internat. Symp. Symbolic Algebraic Comput. ISSAC'03*, pages 215–223, New York, 2003. ACM Press. ISBN 1-58113-641-2.

Wu, X. and Zhi, L. Computing the multiplicity structure from geometric involutive form. In *Proc. 2008 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'08)*, pages 325–332, 2008a.

Wu, X. and Zhi, L. Determining singular solutions of polynomial systems via symbolic-numeric reduction to geometric involutive form. *MM Research Preprints*, 27:104–122, 2008b.

Zhi, L. and Reid, G. Solving nonlinear polynomial system via symbolic-numeric elimination method. In Faugére, J. and Rouillier, F., editors, *Proc. International conference on polynomial system solving*, pages 50–53, 2004. Full version of the paper in J. Symoblic Comput. 44(3), 280-291.

ASCM 2009 Regular Session

Real Root Isolation of Regular Chains

François Boulier[1], Changbo Chen[2], François Lemaire[1], and Marc Moreno Maza[2]

[1] LIFL, Université de Lille 1
59655 Villeneuve d'Ascq Cedex, France
{Francois.Boulier,Francois.Lemaire}@lifl.fr

[2] ORCCA, University of Western Ontario (UWO)
London, Ontario, Canada
{cchen252,moreno}@csd.uwo.ca

Abstract

We present an algorithm RealRootIsolate for isolating the real roots of a system of multivariate polynomials given by a zerodimensional squarefree regular chain. The output of the algorithm is guaranteed in the sense that all real roots are obtained and are described by boxes of arbitrary precision. Real roots are encoded with a hybrid representation, combining a symbolic object, namely a regular chain, and a numerical approximation given by intervals. Our isolation algorithm is a generalization, for regular chains, of the algorithm proposed by Collins and Akritas. We have implemented RealRootIsolate as a command of the module SemiAlgebraicSetTools of the RegularChains library in Maple. Benchmarks are reported.

1 Introduction

Finding real roots for univariate polynomials has been widely studied. Some methods guarantee the number of real roots and isolate each real root in an arbitrary small interval. The algorithm presented in this paper is a generalization to regular chains of the algorithm given by Collins and Akritas [8], whose termination proof is based on a theorem due to Vincent [32] and [24, Theorem of 2 circles].

There exist many different approaches for isolating real roots of univariate polynomials by means of Descartes rules of signs [13]. Uspensky [31] rediscovered independently* an inefficient version of Vincent's work [1]. More recent algorithms are closer to the original work of Vincent and based on continuous fractions [2, 3]. The approach of [29] is very efficient in memory since it avoids the storage of one polynomial at each node of the tree of the recursive calls. Observe that the application of this idea to our context should be possible. It is left to future work.

The methods mentioned above are all for univariate polynomials with integral or rational coefficients. In [14], the authors apply Descartes Algorithm for polynomials with bit-stream coefficients. In [9, 16], the authors present algorithms for isolating the real roots of univariate polynomials with real algebraic number coefficients.

There exist different approaches for isolating real roots of polynomial systems with finitely many complex solutions. Various constructions are employed to generalize to multivariate systems the techniques known for univariate equations: rational univariate representation [27], polyhedron algebra [23], and triangular decompositions [7, 20, 25, 35].

In this paper, we generalize the Vincent-Collins-Akritas Algorithm to zerodimensional squarefree regular chains; therefore our work falls in the same category as this latter group

*Recent investigations of A. Akritas seem to prove that Uspensky only had an incomplete knowledge of Vincent's paper, from [30, pages 363-368].

of papers. Our idea is to build inductively (one variable after another) "boxes" in which one and only one real solution lies. This basically amounts to applying the Vincent-Collins-Akritas Algorithm to polynomials with real algebraic coefficients defined by a regular chain. Our main algorithm RealRootIsolate takes a zerodimensional squarefree regular chain T as an input and returns a list of disjoint boxes (Cartesian products of intervals) such that each box contains exactly one real root of T (as a byproduct, RealRootIsolate counts the number of real roots). We have implemented our algorithm in MAPLE in the module `SemiAlgebraicSetTools` of the `RegularChains` library.

Although rediscovered independently, the techniques presented here share some ideas with those of [25, 26]. However, our algorithm focuses on finding isolation boxes for real solutions of polynomial systems whereas Rioboo's primary goal is to implement the real closure of an ordered field. Moreover, Rioboo relies on Sturm sequences and subresultants for univariate polynomial real root isolation.

The other real root isolation algorithms based on triangular decompositions, namely those reported in [7, 20, 35], rely on the so-called "sleeve polynomials", see Section 2.5.

We do not report on a comparative implementation with the methods in [7, 9, 20, 25, 35]. In order to ensure a fair comparison (similar to what was done in [4] for four triangular decomposition methods) one would need to bring these six real root isolation methods (including ours) in a common implementation framework. This would require a significant amount of work, and we leave this comparative implementation for future work.

As mentioned, the algorithm presented here has been implemented in MAPLE interpreted code. However, it does not rely yet on fast polynomial arithmetic nor modular methods for regular chain computations. Following [18], these techniques should speed-up our implementation dramatically.

We compare our code with another real root isolation tool available in MAPLE: the RootFinding[Isolate] command based on the rational univariate representation [27]. With no surprise, the highly optimized supporting C code allows RootFinding[Isolate] to outperform our modest MAPLE implementation on systems that are in Shape Lemma position [5]. However, for different families of examples, corresponding to non-equiprojectable[†] varieties the situation is reversed which demonstrates the great interest of our approach, even in this unfair comparative implementation framework.

Another contribution of our work is that it equips MAPLE with a tool for manipulating real numbers exactly. For instance, our code provides a data-type (called a *box*) for encoding a point with n coordinates that are real algebraic numbers, together with a function for deciding whether this point cancels a given n-variate polynomial. Our encoding of such points uses a hybrid representation, combining a symbolic object, namely a regular chain, and a numerical approximation given by intervals.

We investigate the impact of different strategies for isolating roots. In particular, we identify a family of examples where the use of normalized regular chains instead of arbitrary (but still zero-dimensional) regular chains can speed-up the root isolation even though normalization tends to substantially increase coefficient sizes, as established in [11].

2 Real root isolation of a zerodimensional regular chain

After recalling the Vincent-Collins-Akritas algorithm in Section 2.1 and introducing definitions in Section 2.2 and Section 2.3, the algorithm RealRootIsolate and its subalgorithms

[†]The notions of an equiprojectable variety and equiprojectable decomposition are discussed in [10].

are presented in Section 2.4. In Section 2.5 we compare our method with other existing approaches.

2.1 The Vincent-Collins-Akritas algorithm

The Vincent-Collins-Akritas algorithm isolates the real roots of a squarefree polynomial (with rational coefficients) with an arbitrary precision. A basic version (Algorithm 1) is recalled here, as a first step to its generalization in Section 2.4.

Definition 1 *Let V be a finite set of t real numbers. An* interval decomposition *of V is a list $I_1, \ldots, I_t$ of intervals such that each interval I_i is an open rational interval $]a, b[$ or a rational singleton $\{a\}$, each interval I_i contains one element of V and $I_i \cap I_j = \emptyset$ if $i \neq j$.*

Algorithm 1 RootIsolateVCA(p)

Input: p squarefree polynomial of $\mathbb{Q}[x]$
Output: an interval decomposition of the real roots of p
1: $H \leftarrow$ a strict bound on the roots of p
2: **return** RootIsolateAuxVCA($p,]-H, H[$)

In Algorithm 1, there are different ways to compute a strict bound H (in the sense that any root α of p satisfies $|\alpha| < H$). For example, if $p = \sum_{i=0}^{d} a_i x^i$, take the Cauchy bound $H = \frac{1}{|a_d|} \sum_{i=0}^{d} |a_i|$. Sharper bounds are given in [2].

Algorithm 2 RootIsolateAuxVCA($p,]a, b[$)

Input: p squarefree polynomial in $\mathbb{Q}[x]$ and $a < b$ rational
Output: an interval decomposition of the real roots of p which lie in $]a, b[$
1: $nsv \leftarrow$ BoundNumberRootsVCA($p,]a, b[$)
2: **if** $nsv = 0$ **then return** $\emptyset$
3: **else if** $nsv = 1$ **then return** $]a, b[$
4: **else**
5: $\quad m \leftarrow (a+b)/2 \quad res \leftarrow \emptyset$
6: $\quad$ **if** $p(m) = 0$ **then** $res \leftarrow \{\{m\}\}$
7: $\quad$ {Next line ensures the roots are sorted increasingly}
8: $\quad$ **return** RootIsolateAuxVCA($p,]a, m[$) $\cup$ res $\cup$ RootIsolateAuxVCA($p,]m, b[$)

Algorithm 3 BoundNumberRootsVCA($p,]a, b[$)

Input: $p \in \mathbb{Q}[x]$ and $a < b$ rational
Output: a bound on the number of roots of p in the interval $]a, b[$
1: $\bar{p} \leftarrow (x+1)^d \, p\left(\frac{a\,x+b}{x+1}\right)$ where d is the degree of p, and denote $\bar{p} = \sum_{i=0}^{d} a_i x^i$
2: $a'_e, \ldots, a'_0 \leftarrow$ the sequence obtained from $a_d, \ldots, a_0$ by removing zero coefficients
3: **return** the number of sign variations in the sequence $a'_e, \ldots, a'_0$

The main arguments for the correctness of Algorithm 1 are the following. Algorithm 3 computes a polynomial $\bar{p}$ whose positive real roots are in bijection with the real roots of p

which lie in $]a, b[$. The application of Descartes' rule of signs on $\bar{p}$ thus provides a bound on the number of real roots of p which lie in $]a, b[$. This bound is exact when equal to 0 or 1 [24, Theorem 1.2]. Since p is squarefree, the bound returned by Algorithm 3 will eventually become 0 or 1, by [24, Theorem 2.5] so that the whole method terminates.

2.2 Regular chains

In this paper one only considers zerodimensional squarefree regular chains, abbreviated by *zs-rc*. Roughly speaking, a zerodimensional regular chain is a triangular set[‡] of polynomials, with as many equations as variables, and which has a finite number of complex roots (and consequently a finite number of real roots).

Zerodimensional regular chains are easier to understand and define than general regular chains. Let $x_1 < \cdots < x_s$ be s variables. Let $p \in \mathbb{Q}[x_1, \ldots, x_s]$ be a non-constant polynomial. We denote by $\mathrm{mvar}(p)$ the *main variable* (or largest variable) of p, by $\mathrm{init}(p)$ the *initial* (or leading coefficient w.r.t. $\mathrm{mvar}(p)$) of p, by $\mathrm{mdeg}(p)$ the degree of p in its main variable and by $\mathrm{sep}(p)$ the *separant* of p, that is $\partial p / \partial \mathrm{mvar}(p)$. If T is a set of polynomials in $\mathbb{Q}[x_1, \ldots, x_s]$, $\langle T \rangle$ denotes the ideal generated by T and $V(T)$ denotes the set of all complex solutions of the system $T = 0$. For a given x_i, $T_{\leq x_i}$ (resp. $T_{>x_i}$) denotes the elements of T whose main variable is less (resp. strictly greater) than x_i.

Definition 2 *Let $T = \{p_1, \ldots, p_s\}$ where each p_i lies in $\mathbb{Q}[x_1, \ldots, x_s]$. The set T is a* zerodimensional squarefree regular chain *(or* zs-rc*) of $\mathbb{Q}[x_1, \ldots, x_s]$ if $\mathrm{mvar}(p_i) = x_i$ for $1 \leq i \leq s$, $\mathrm{init}(p_i)$ does not vanish on $V(\{p_1, \ldots, p_{i-1}\})$ for any $2 \leq i \leq s$, and $\mathrm{sep}(p_i)$ does not vanish on $V(\{p_1, \ldots, p_i\})$ for any $1 \leq i \leq s$*

Thanks to the first two conditions, it is easy to show that the system $T = 0$ has a finite number of complex solutions, which is equal to the product of the main degrees of the elements of T denoted $\textsc{Deg}(T)$. Moreover those solutions can be computed "incrementally" using the following solving scheme. The number of complex roots of the univariate polynomial p_1 is equal to its degree. For each root x_1^0 of p_1, consider the polynomial $p_2(x_1^0, x_2)$ which is univariate in x_2. This polynomial has the same degree in x_2 as p_2 since the initial of p_2 does not vanish on the solutions of $p_1 = 0$. Thus, the number of complex roots of $p_2(x_1^0, x_2)$ is equal to the degree of p_2. Proceeding on the remaining variables, one concludes that the number of complex solutions of $T = 0$ is equal to $\textsc{Deg}(T)$.

The third condition, which forbids multiple roots, is the natural generalization of squarefree polynomials to regular chains. As for the algorithm RootIsolateVCA, this condition is only required to make the isolation algorithms terminate.

In practice, the zs-rc can be computed using the Triangularize algorithm [22] available in the `RegularChains` library shipped with Maple. Moreover, the regular chains are not built by checking the conditions of Definition 2 but by using regularity tests of polynomials modulo ideals. A polynomial p is said to be *regular* modulo an ideal I if it is neither zero, nor a zero-divisor modulo I. If T is a regular chain, p is said to be regular modulo the regular chain T if p is regular modulo $\langle T \rangle$. Thus, the following definition is equivalent to Definition 2.

Definition 3 *Let $T = \{p_1, \ldots, p_s\}$ where each p_i lies in $\mathbb{Q}[x_1, \ldots, x_s]$. The set T is a* zerodimensional squarefree regular chain *(or* zs-rc*) of $\mathbb{Q}[x_1, \ldots, x_s]$ if $\mathrm{mvar}(p_i) = x_i$ for any $1 \leq i \leq s$, $\mathrm{init}(p_i)$ is regular modulo the ideal $\langle p_1, \ldots, p_{i-1} \rangle$ for any $2 \leq i \leq s$, and $\mathrm{sep}(p_i)$ is regular modulo the ideal $\langle p_1, \ldots, p_i \rangle$ for any $1 \leq i \leq s$.*

[‡]triangular set in the sense that each polynomial introduces exactly one more variable

The expert reader has probably noticed that saturated ideals are not mentioned. Indeed, this precision in not necessary in dimension zero. The next lemma makes the link between the regularity property of a polynomial q modulo a zs-rc and the fact that q does not vanish on the solutions of a zs-rc. It is implicitly used to check whether or not a polynomial vanishes on a root of a regular chain in the CheckZeroDivisor algorithm.

Lemma 1 *Let T be a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$ and q a polynomial of $\mathbb{Q}[x_1, \ldots, x_s]$. Then q is regular modulo T iff q does not vanish on any complex solution of T.*

2.3 Boxes

This section defines the boxes used for isolating solutions of zs-rc, as well as extra definitions needed to specify the algorithms of Section 2.4.

Definition 4 *An s-box (or box) B is a Cartesian product $B = I_1 \times \cdots \times I_s$ where each I_i is either a rational open interval $]a, b[$ (a and b are rational) or a singleton $\{a\}$ with a rational. The width of B, denoted by $|B|$, is defined as the maximum of the $|I_i|$ where $|I_i| = 0$ if it is a singleton and $b - a$ if $I_i =]a, b[$.*

Algorithm 4 EvalBox(p, B)

Input: $p \in \mathbb{Q}[x_1, \ldots, x_s]$ and B is a s-box
Output: a rational interval I such that $p(v) \in I$ for any $v \in B$

Different variants for EvalBox(p, B) exist. A simple version of this algorithm consists in using the three basic operations $(\times, -, +)$ defined in interval arithmetics after "expanding" p into a sum of monomials. Any variant for EvalBox(p, B) satisfying the following property can be used: the box EvalBox(p, B) should tend to the singleton $\{p(x_0)\}$ when the width of B tends to zero (by keeping the condition $x_0 \in B$). This simply ensures that the interval EvalBox(p, B) should shrink as the width of the box B decreases.

Definition 5 *Let $B = I_1 \times \cdots \times I_s$ be an s-box and $T = \{p_1, \ldots, p_s\}$ be a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$. We say (B, T) satisfies the Dichotomy Condition (or **DC**) if*

- *one and only one real root of T lies in B*
- *if $I_1 =]a, b[$ then $p_1(x_1 = a)$ and $p_1(x_1 = b)$ are nonzero and have opposite signs*
- *for each $2 \leq k \leq s$, if $I_k =]a, b[$ then the two intervals EvalBox$(p_k(x_k = a), B)$ and EvalBox$(p_k(x_k = b), B)$ do not meet 0 and have opposite signs.*[§]

This last condition is the natural generalization of the condition $p(a)$ and $p(b)$ are nonzero and have opposite sign, and p vanishes only once on the interval $]a, b[$ in the univariate case. Condition **DC** allows to refine a box very much like one refines the interval $]a, b[$ by dichotomy.

Please note that if $I_1 = a$ is a singleton, then one necessarily has $p_1(a) = 0$ since one real root of T lies in B. Equivalently, if $I_k = a$ is a singleton for some k, then $p(x_k = a)$ vanishes on the real root of T lying in B.

[§] the sign of an interval not meeting zero is just the sign of any element of it

Definition 6 *Let V be a finite set of t points of $\mathbb{R}^s$. A list $B_1, \ldots, B_t$ of s-boxes is called a* box-decomposition *of V if each point of V lies in exactly one B_i and $B_i \cap B_j = \emptyset$ if $i \neq j$. If T is a zs-rc, we call box-decomposition of T a box-decomposition of the real roots of $T = 0$.*

Definition 7 *A task $\mathcal{M} = \text{TASK}(p,]a, b[, B, T)$ is defined as: T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$, p is a polynomial in $\mathbb{Q}[x_1, \ldots, x_{s+1}]$, $T \cup \{p\}$ is a zs-rc, B is an s-box, (B, T) satisfies **DC**, and $a < b$ are rational numbers. The solution of $\mathcal{M}$ denoted by $V_t(\mathcal{M})$ is defined as $V(T \cup \{p\}) \cap (B \times]a, b[)$ (i.e. the real solutions of $T \cup \{p\}$ which prolong the real root in B and whose component x_{s+1} lies in $]a, b[$).*

2.4 Algorithms

The main algorithm RealRootIsolate, which isolates the real roots of a zerodimensional squarefree regular chain, is presented here. Only elements of proofs are given but focus has been made on specifications. In this section, one assumes $n > 1$.

The algorithms presented here use the mechanism of exceptions which is available in a lot of programming languages. We find it appropriate since doing computations using the D5 principle [12] can be seen as doing computations as if one is computing over a field. When a zero divisor is hit (leading to a splitting), one raises an exception exhibiting the splitting. This exception can then be caught to restart computations. This shortens and makes clearer[¶] the algorithms presented here. Only the Algorithm 5 throws exceptions.

Algorithm 5 CheckZeroDivisor(p, T)

Input: T a zs-rc $\mathbb{Q}[x_1, \ldots, x_s]$ and $p \in \mathbb{Q}[x_1, \ldots, x_s]$
Output: If p is regular modulo T, then the algorithm terminates normally. Otherwise, an exception is thrown exhibiting t zs-rc $T_1, \ldots, T_t$ such that **C1** $V(T_1) \cup \cdots \cup V(T_t) = V(T)$, and **C2** $\sum_{i=1}^{t} \text{DEG}(T_i) = \text{DEG}(T)$ hold.

1: $T_1, \ldots, T_t \leftarrow$ Regularize(p, T)
2: **if** p belongs to at least one $\langle T_i \rangle$ **then throw exception**$(T_1, \ldots, T_t)$

Algorithm 5 checks whether p is regular modulo T or not. If p is regular modulo T, the algorithm returns normally, otherwise an exception is raised. Algorithm 5 is called whenever one wants to know if a polynomial vanishes on a real root x^0 of T isolated by a box B. Indeed, if p is regular modulo T, thanks to Lemma 1, p does not vanish on x^0. This allows to refine B until EvalBox(p, B) does not contain 0, which gives the sign of $p(x^0)$.

The algorithm Regularize is not recalled here (see [22] for details) but its specification is: if T is a zs-rc, Regularize(p, T) returns a list of zs-rc $T_1, \ldots, T_t$ such that for each T_i, p either belongs to $\langle T_i \rangle$ or is regular modulo T_i. Moreover $T_1, \ldots, T_t$ is (what we call) a splitting of T, which in dimension 0 satisfies the two conditions **C1** and **C2** of the output of Algorithm 5. Due to condition **C2**, splittings cannot occur indefinitely.

Algorithm 6 RefineBox(B, T)

Input: T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$, (B, T) satisfies **DC** and $|B| > 0$
Output: an s-box $\bar{B}$ such that $|\bar{B}| \leq |B|/2$, $\bar{B} \subset B$ and $(\bar{B}, T)$ satisfies the **DC**

Algorithm 6 is able to refine a box containing a real root by dividing its width by 2. It is simply the generalization of the dichotomy process for splitting in two an isolating interval.

[¶] without using exceptions, splittings need to be handled basically each time a function returns a value

The algorithm is not detailed here for brevity. The main idea is to divide by two each interval I_i of $B = I_1 \times \cdots \times I_s$ which is larger than $|B|/2$ while keeping the **DC** condition.

Algorithm 7 RealRootIsolate(T)

Input: T is a zs-rc
Output: a box-decomposition $B_1, \ldots, B_p$ of T

1: $I_1, \ldots, I_t \leftarrow \mathsf{RootIsolateVCA}(T_{x_1})$
2: $toDo \leftarrow \{(T_{>x_1}, (I_i, T_{\leq x_1}))\}_{1 \leq i \leq t}$
3: $res \leftarrow \emptyset$
4: **while** $toDo \neq \emptyset$ **do**
5: pick and remove a $(T_{>x_i}, (B, T_{\leq x_i}))$ from $toDo$
6: $B'_1, \ldots, B'_{t'} \leftarrow \mathsf{SolveNewVar}(T_{x_{i+1}}, B, T_{\leq x_i})$
7: **if** $x_{i+1} = x_n$ **then** $res \leftarrow res \cup \{B'_1, \ldots, B'_{t'}\}$
8: **else** $toDo \leftarrow toDo \cup \{(T_{<x_{i+1}}, (B'_j, T_{\geq x_{i+1}}))\}_{1 \leq j \leq t'}$
9: **return** res

Algorithm 7 is a generalization of Algorithm 1 for a zs-rc. Line 1 isolates the real roots of the univariate polynomial T_{x_1}. The variable $toDo$ is a set of $(T_{>x_i}, (B, T_{\leq x_i}))$ such that each $(B, T_{\leq x_i})$ satisfies **DC**. It means that $(B, T_{\leq x_i})$ represents one (and only one) real root of $T_{\leq x_i}$. The set $T_{>x_i}$ simply is the set of polynomials which have not be solved yet. Algorithm 7 calls Algorithm 8 (which allows to solve one new variable) until all variables are solved. Note that Algorithm 7 could be followed by a refinement of each returned box so that the width of each box is smaller than a given precision.

Also remark that any raised exception will hit Algorithm 7 since none of the algorithms presented here catches any exception. It is however very easy to adjust Algorithm 7 so that it would catch exceptions and recall itself on each regular chain returned by the splitting. The recursion would eventually stop because of condition **C2** of Algorithm 5 (i.e. splittings cannot occur indefinitely).

Algorithm 8 SolveNewVar(p, B, T)

Input: T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$, $p \in \mathbb{Q}[x_1, \ldots, x_{s+1}]$ and $T \cup \{p\}$ is a regular chain and (B, T) satisfies **DC**
Output: a box-decomposition of the roots $(x_1^0, \ldots, x_{s+1}^0)$ of $T \cup \{p\}$ such that $(x_1^0, \ldots, x_s^0)$ is the root of T which lies in B

1: refine B into a box B' such that $0 \notin \mathsf{EvalBox}(i_p, B')$
2: compute a bound H on the roots of $p(x_1^0, \ldots, x_s^0, x_{s+1})$
3: $toDo \leftarrow \{\textsc{task}(p,] - H, H[, B', T)\}$
4: $res \leftarrow \emptyset$
5: **while** $toDo \neq \emptyset$ **do**
6: pick and remove a task $\mathcal{M}$ from $toDo$
7: **for all** e in $\mathsf{SolveTask}(\mathcal{M})$ **do**
8: **if** e is a box **then** $res \leftarrow res \cup \{e\}$ **else** $toDo \leftarrow toDo \cup \{e\}$
9: **return** res

Algorithm 8 finds the real roots of p (seen as univariate in x_{s+1}) that "prolong" the real root which lies in B. Line 1 always terminates. Indeed, i_p is regular modulo T, so it does not vanish on any root of T. Thus, ultimately, B' will be small enough so that $0 \notin \mathsf{EvalBox}(i_p, B')$. Refining the box is needed to compute the bound H.

The bound H at line 2 can be computed in the following way. Denote $p = \sum_{i=0}^{d} a_i x_{s+1}^i$ and $A_i = \mathsf{EvalBox}(a_i, B')$. Then take $H = \frac{1}{\min|A_d|} \sum_{i=0}^{d} (\max|A_i|)$ where $\min|A_i|$ (resp. $\max|A_i|$) denotes the minimum (resp. maximum) of the modulus of the bounds of the interval A_i. The rest of the algorithm is based on Algorithm 9 which transforms tasks into new tasks and boxes.

Algorithm 9 $\mathsf{SolveTask}(\mathcal{M})$

Input: a task $\mathcal{M} = \textsc{task}(p,]a, b[, B, T)$ where T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$

Output: One of the four following cases:

1: $\emptyset$ which means $V_t(\mathcal{M}) = \emptyset$.

2: a box B' such that $(B', T \cup \{p\})$ satisfies **DC** and B' is a box-decomposition of $V_t(\mathcal{M})$, which means $V_t(\mathcal{M})$ is composed of only one point

3: two tasks $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $V_t(\mathcal{M}_1)$ and $V_t(\mathcal{M}_2)$ forms a partition of $V_t(\mathcal{M})$

4: two tasks $\mathcal{M}_1$ and $\mathcal{M}_2$ plus a box B' such that $(B', T \cup \{p\})$ satisfies **DC** and the three sets $V_t(\mathcal{M}_1)$, $V_t(\mathcal{M}_2)$ and $\{x^0\}$ form a partition of $V_t(\mathcal{M})$, where x^0 denotes the only real root of $T \cup \{p\}$ which lies in B'.

1: $nsv, B' \leftarrow \mathsf{BoundNumberRoots}(\mathcal{M})$
2: **if** $nsv = 0$ **then return** $\emptyset$
3: **else if** $nsv = 1$ **then**
4: $\quad B'' \leftarrow B' \times]a, b[$
5: $\quad$ refine B'' until $(B'', T \cup \{p\})$ satisfies **DC**
6: $\quad$ **return** $\{B''\}$
7: **else**
8: $\quad m \leftarrow (a+b)/2 \quad res \leftarrow \emptyset \quad p' \leftarrow p(x_{s+1} = m)$
9: $\quad$ **if** $p' \in \langle T \rangle$ **then** $res \leftarrow \{B' \times \{m\}\}$ **else** $\mathsf{CheckZeroDivisor}(p', T)$
10: $\quad$ **return** $res \cup \{\textsc{task}(p,]a, m[, B', T), \textsc{task}(p,]m, b[, B', T)\}$

Algorithm 10 $\mathsf{BoundNumberRoots}(\mathcal{M})$

Input: a task $\mathcal{M} = \textsc{task}(p,]a, b[, B, T)$ where T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$

Output: (nsv, B') such that $B' \subset B$, (B', T) satisfies **DC**, and nsv is a bound on the cardinal of $V_t(\mathcal{M})$. The bound is exact if $nsv = 0$ or 1.

1: $\bar{p} \leftarrow (x_{s+1}+1)^d \, p\left(x_{s+1} = \frac{a\, x_{s+1} + b}{x_{s+1}+1}\right)$ with $d = \mathrm{mdeg}(p)$
2: denote $\bar{p} = \sum_{i=0}^{d} a_i x_{s+1}^i$
3: $a'_e, \ldots, a'_0 \leftarrow$ the sequence obtained from $a_d, \ldots, a_0$ by removing the a_i belonging to $\langle T \rangle$
4: **for all** a'_i **do** $\mathsf{CheckZeroDivisor}(a'_i, T)$
5: $B' \leftarrow B$
6: **while** there is an a'_i such that $0 \in \mathsf{EvalBox}(a'_i, B')$ **do** $B' = \mathsf{RefineBox}(B', T)$
7: **return** the number of sign variations of the sequence
$\quad \mathsf{EvalBox}(a'_e, B'), \mathsf{EvalBox}(a'_{e-1}, B'), \ldots, \mathsf{EvalBox}(a'_0, B')$

Algorithm 9 is a generalization of Algorithm 2. The cases $nsv = 0$ or 1 are straightforward. When $nsv > 1$, one needs to split the interval $]a, b[$ in two, yielding two tasks returned on line 10. Lines 8-9 correspond to the lines 5-6 of Algorithm 2. Indeed, checking $p(m) = 0$ is transformed into checking if p' lies in $\langle T \rangle$ or is not a zero divisor modulo T.

Algorithm 10 is a generalization of Algorithm 3. One discards the coefficients of p' which lie in $\langle T \rangle$ because they vanish on the real root v which is in B. One also ensures that

the other coefficients (the a'_i) are not zero divisors, so they cannot vanish on v. Thus the loop at line 6 terminates. Moreover, this guarantees that the number of sign variations is correct. Please note that the sequence $a'_e, \ldots, a'_0$ is never empty. Indeed if all a_i's were in $\langle T \rangle$, then all coefficients of p would lie in $\langle T \rangle$ (impossible since i_p is regular modulo T).

2.5 Comparison with other methods

In the introduction we provided a comparison of our work with others. More technical details are reported below.

[25, 26] give algorithmic methods (available in AXIOM) to manipulate real algebraic numbers. These developments were designed for improving *Cylindrical Algebraic Decomposition* (CAD) methods in AXIOM. Although [25] contains all the tools to solve our problem, this paper focuses on the problem of manipulating real algebraic numbers. It does not address directly the problem of isolating the real roots of a given zerodimensional regular chain. [26] provides tools to perform univariate polynomial real root isolation by using *quasi Sylvester sequence* which according to [26] can be faster than the techniques based on the Descartes rules.

[9, 16] present algorithms for isolating real roots of univariate polynomials with algebraic coefficients. Their algorithms require the ideal to be prime, and this condition is ensured by performing univariate factorization [21] into irreducible factors for polynomials with algebraic coefficients. Our method does not require such factorizations and only requires the ideal to be squarefree. Thus, our method replaces a decomposition into prime ideals by regularity tests which are often less costly.

[27] is based on Gröbner basis computations and rational univariate representation. Thus, [27] transforms the initial problem into the problem of isolating the real roots of a univariate polynomial with rational number coefficients

[20] starts from a zerodimensional regular chain (although [20] uses the terminology of characteristic sets) and proceeds variable by variable. Their technique is different from ours. After isolating a real root say x_1^0 for $p_1(x_1) = 0$, they build two univariate polynomials $\overline{p}_2(x_2)$ (the so-called upper bound polynomial) and $\underline{p}_2(x_2)$ (the so-called lower bound polynomial) whose real roots will interleave nicely (see [20, Definition 2]) when the precision on x_1^0 is sufficiently low, yielding isolation intervals for the variable x_2.

[35] uses a similar techniques as [20]. The main difference is that the authors use explicitly interval arithmetic and contrarily to [20] where the algorithm may have to restart from the beginning with a smaller precision (called *ac*) in some special cases, [35] uses a refinement process (algorithm NSHR) until the real roots of the upper and lower bound polynomial interleave sufficiently.

Such techniques are also used in [7], where the authors consider general zerodimensional triangular systems (which may not be a regular chain) and treat multiple zeros directly.

Quoting the abstract of [23], the Authors use *a powerful reduction strategy based on univariate root finder using Bernstein basis representation and Descartes' rule*. Basically, they reduce the problem to solving univariate polynomials by using the Bernstein basis representation and optimizations based on convex hulls.

3 Implementation

3.1 The SemiAlgebraicSetTools package

The algorithm RealRootIsolate has been coded using exceptions in MAPLE in the module `SemiAlgebraicSetTools` of the `RegularChains` library [17]. We present some implementation issues and optimizations integrated in our code.

Precision. The user can specify a positive precision so all isolation boxes have a width smaller than the given precision. If an infinite precision is provided, then the algorithm only isolates the real roots by refining the boxes the least possible. We take the precision into account as soon as possible in the algorithm, meaning that each time an isolation box is extending with a new variable, one refines the box.

Constraints. The user can restrict the solutions by imposing that some variables lie in a prescribed interval. If the intervals are restrictive (i.e. smaller than the intervals computed using bounds), this helps avoiding useless computations.

The CheckZeroDivisor algorithm is not directly called in our code. Indeed, regularity test can be very expensive and should be avoided as much as possible. When a call CheckZeroDivisor(p, T) returns, one knows that a box B isolating a real root of T can always be refined until the interval EvalBox(p, B) does not meet zero. This is in fact the only reason why we call CheckZeroDivisor. In order to avoid a regularity test, we first try to refine B a few times to see if EvalBox(p, B) still meets zero. If it does not, we do not need to check the regularity.

Refining boxes. In the MAPLE implementation, Algorithm 6 receives an extra parameter x_k. In that case, the box is only refined for the variables smaller than x_k (i.e. the variables x_i with $i \leq k$). This is useful for example at line 6 of Algorithm BoundNumberRoots. Indeed, if $\mathrm{mvar}(a'_i) = x_k$ holds, then it is not necessary to refine the complete box B' to ensure that EvalBox(a'_i, B') does not meet 0.

Change of variables. By slightly modifying algorithms 8 and 9, we call algorithm 10 with $a = 0$ and $b = 1$. This allows to replace the operation $(x_{s+1}+1)^d\, p\left(x_{s+1} = \frac{a\, x_{s+1}+b}{x_{s+1}+1}\right)$ by substitutions of the form $p(x_{s+1} = x_{s+1}/2)$, $p(x_{s+1} = 1/x_{s+1})$ and $p(x_{s+1} = x_{s+1}+1)$ which can be written very efficiently, the last one being based on fast Taylor shift [33].

Refining other branches. Due to the triangular structure of the system, many different roots share a common part (meaning the values for some variables are equal). When refining a root, we refine other roots which share a common part to save computations.

Further refining. After being computed, an isolation box isolating a real root v can be refined further using the MAPLE command RefineBox. To do so, exceptions has to be caught. Our implementation associates a regular chain T to each box B encoding a real root. Thus, if T is split into $T_1, \ldots, T_s$ after an exception, one replaces (B, T) by the right (B, T_i) which also defines the real root v as done in [26, page 528].

EvalPoly. For evaluating EvalBox(p, B), we first collect p using a Hörner scheme. For example, the polynomial $p := x_2^3x_1 + 3x_2^2 + x_2x_1^2 + x_1^2 + x_1 + 1$ is collected as $1 + (1 + x_1)x_1 + (x_1^2 + (3 + x_1x_2)x_2)x_2$. This strategy seems to behave quite well on our examples. The intuition for doing that is the following. Since $x_2 > x_1$ for our ordering, the interval of B for the variable x_2 tend to be in practice wider than that for the variable x_1, since the intervals for smaller variables tend to be more refined than those for higher variables. On the example, the Hörner collected form tends to decrease the exponentiations for x_2.

3.2 Further development

Using fast polynomial arithmetic and modular methods. The current implementation of the CheckZeroDivisor algorithm can be improved in a significant manner. Indeed, the modular algorithm for regularity test of [18] and implemented with the MODPN library [19] outperform the regularity test used in CheckZeroDivisor by several orders of magnitude.

Computing with algebraic numbers. Using the two algorithms RefineBox and CheckZeroDivisor, one can encode algebraic numbers and check if a multivariate polynomial cancels on some algebraic numbers. This allows computing with algebraic numbers, very much as in [25]. Moreover, inequations and inequalities could be included with almost no work. Indeed they can be handled at the end of RealRootIsolate using CheckZeroDivisor. They can also be treated inside the subalgorithms as soon as a box in construction involves all the variables of an inequality or inequation, allowing to cut some branches.

Floating-point computations. As suggested by Fabrice Rouillier (private communication), it would speed up the algorithm to use multiple-precision floating-point computations with exact rounding (as in the MPFI library [28]) instead of rational numbers.

Exceptions could be caught sooner so one does not lose the computations already done.

Unsing continuous fractions as in [2, 3] may also be investigated.

Interval arithmetics. The algorithm EvalBox could certainly be improved by techniques such as [6] where the polynomial to evaluate is factorized using greedy algorithms.

Newton method. Some tries were made to incorporate a Newton method for system of polynomials in the RefineBox algorithm. Due to the triangular form of the system, the jacobian is also triangular which eases the method. However, although the convergence was really faster, it was not satisfactory because of the coefficient swell of the isolation intervals. However, we believe the Newton method should be investigated more carefully.

4 Benchmarks

4.1 Description of the experimentation

The names of the examples used for benchmarking are listed in Figure 4.3. Most of them are classical. The lhlp files tests are taken from [20]. The examples *chemical-reaction*, *geometric-constraints*, *neural-network*, *p3p-special* and *Takeuchi-Lu* appear in [34]. The nld-*d-n* and nql-*n-d* examples are described in Section 4.3. Examples can be found at `www.lifl.fr/~lemaire/BCLM09/BCLM09-systems.txt`.

Benchmark results are given on Figure 4.3. They were run on an Intel(R) Pentium(R) D CPU 3GHz with 2Gb of memory, using MAPLE 13 beta 64bits. Timings are in seconds. Timeouts are indicated with >, meaning that the computation was aborted. The column *Sys* denotes the name of the system. The column *v/e/s* stands for the number of variables/equations/real solutions.

The Maple command RootFinding[Isolate] isolates real roots within the times indicated in the group of columns RF/Is. For multivariate systems, this command relies on Gröbner basis computations [15] and rational univariate representation [27]. In Column 1, the command used is RootFinding [Isolate](sys, variables, digits=10, output=interval). For Column 2 the same command is used but that the ordering of the variables has been reversed. We used those two commands in case the variable ordering has an effect on the command RootFinding[Isolate]. Note that the option digits=10 ensures that the ten first digits of the results are correct which is not the same as guaranteeing a width less than `1e-10` for the isolation

boxes in RealRootIsolate. However, the difficulty for isolating the real roots is comparable if the real roots are not too close to zero nor too big; this is the case for our test examples.

The other groups of columns correspond to three strategies for isolating real roots using our algorithm RealRootIsolate. In each strategy, the initial system is first decomposed into zerodimensional regular chains using the Triangularize command together with the option radical='yes' ensuring those regular chains are squarefree. In order to keep things simple and uniform, the option probability=xx of Triangularize is not used, even when it could be, that is, for square systems generating radical ideals. Therefore the modular algorithm of [10] is not applied even though it can solve all our examples that the non-modular algorithm of Triangularize cannot.

Strategy 1. We build regular chains (column Tr) and call the RealRootIsolate algorithm (column Is/10) on each regular chain with a precision of `1e-10`.

Strategy 2. A variant of Strategy 1 where we compute strongly normalized regular chains (column Tr/No) using the option normalized='strongly' of Triangularize.

Strategy 3. Another variant of Strategy 1. We build regular chains (column Tr) and call the RealRootIsolate algorithm on each regular chain with an infinite precision (column Is/∞), in the sense that the width of the boxes are not constrained. Thus, only the isolation is performed. Then we call the command RefineListBox to refine the list of boxes with a precision of `1e-5` (column ∞/5). Then we refine again the boxes for a precision of `1e-10` (column 5/10).

4.2 Comparison of different strategies

Strategies 1 and 2 are comparable. Strongly normalized regular chains take more time to be computed, since normalization is a post-processing for the command Triangularize. The isolation time is roughly the same in general for both types of regular chains. For the nld-*d*-*n* (except nld-9-3) family of examples, normalization helps the isolation process. However, for some other examples, such as 5-body-homog, p3p-special and Rose, normalization make things worse.

Compared to Strategy 1, Strategy 3 shows two things. First, it is usually faster to isolate solutions with an infinite precision rather than with a small precision. Secondly, it shows that the overall times for Strategies 1 and 3 are comparable.

4.3 Comparison with RootFinding

The RootFinding[Isolate] is obviously a lot faster on many examples. One should keep in mind that this command calls internal routines written in C that have been developed intensively for years. However, the RootFinding[Isolate] has difficulties on some systems such as the nql-*n*-*d* and nld-*d*-*n* ones.

The nql-*n*-*d* (for non quasi linear) example is very specific and was suggested by Fabrice Rouillier. It is defined by n equations in n variables $x_1^d - 2 = 0$, $x_i^d + x_i^{d/2} - x_{i-1} = 0$ for $2 \leq i \leq n$ for some even degree d. This system is already a zs-rc. The algorithm RealRootIsolate solves it easily since the degrees are distributed evenly among the equations. On the other hand, the RootFinding[Isolate] needs to build a rational univariate representation which we believe has a very large degree roughly equal to d^n (that is about one million when $d = 4$ and $n = 10$).

A similar example is simple-nql-*n*-*d* defined by $x_1^d - 2 = 0$, $x_i^d - x_{i-1} = 0$ for $2 \leq i \leq n$. The degree of the rational univariate representation is also roughly d^n. For the example simple-nql-20-30, d^n is around 10^{29}.

		RF/Is		Strategy 1		Strategy 2		Strategy 3			
Sys	v/e/s	1	2	Tr	Is/10	Tr/No	Is/10	Tr	Is/∞	∞/5	5/10
4-body-homog	3/3/7	0.31	0.32	1.6	11	6.2	11	1.5	3.4	4	4.1
5-body-homog	3/3/11	0.31	0.36	3.1	32	38	43	3.2	9.4	11	12
Arnborg-Lazard-rev	3/3/8	<0.1	<0.1	0.43	7	0.5	6.6	0.38	1.8	3	2.6
Arnborg-Lazard	3/3/8	<0.1	<0.1	0.46	7.3	0.62	6.4	0.53	2	3.1	3
Barry	3/3/2	<0.1	<0.1	<0.1	1.5	0.11	3.8	<0.1	0.19	0.64	0.51
Caprasse-Li	4/4/18	0.13	0.14	1.2	3.1	1.4	1.7	1.1	0.44	1.4	1.2
Caprasse	4/4/18	0.13	0.12	1.2	2.9	1.5	2	1.2	0.52	1.6	1.4
chemical-reaction	4/4/4	<0.1	<0.1	0.11	2.8	0.16	2.2	0.12	0.46	1.7	1.1
circles	2/2/22	0.89	0.9	0.55	26	1.1	26	0.59	16	4.6	4.5
cyclic-5	5/5/10	0.4	0.4	2.4	4.6	3.6	1.4	2.5	0.67	3.9	1.8
Czapor-Geddes-Wang	5/5/2	<0.1	0.13	3	6	18	7.9	2.8	2.3	2.5	2.1
fabfaux	3/3/3	<0.1	<0.1	2.4	7.3	51	8.5	2.5	2	3.2	3.3
geom-constraints	3/3/8	<0.1	<0.1	<0.1	2.2	<0.1	2.2	<0.1	0.27	1.2	0.88
GonzalezGonzalez	3/3/2	<0.1	<0.1	<0.1	0.76	0.13	0.75	0.1	0.15	0.4	0.36
Katsura-4	5/5/12	<0.1	<0.1	0.54	14	0.77	19	0.58	2.9	6.8	6.5
lhlp1	3/3/6	<0.1	<0.1	<0.1	1	<0.1	1.5	<0.1	0.14	0.53	0.39
lhlp2	3/3/2	<0.1	<0.1	<0.1	0.95	<0.1	1.4	<0.1	0.19	0.51	0.38
lhlp3	3/3/2	<0.1	<0.1	<0.1	0.63	<0.1	0.75	<0.1	<0.1	0.29	0.24
lhlp4	2/2/4	<0.1	<0.1	<0.1	2.9	<0.1	4.8	<0.1	0.48	1.5	1
lhlp5	3/3/4	<0.1	<0.1	0.26	2	0.26	2.4	0.22	0.35	0.95	0.76
lhlp6	4/4/4	<0.1	<0.1	0.26	2.4	0.34	1.7	0.23	0.36	1.5	0.78
neural-network	4/4/22	1	1	0.81	18	1.2	15	0.87	4.5	7.7	7
nld-3-4	4/4/27	1.1	1.1	3.4	13	4.3	5.1	3.2	2.2	6.6	5.8
nld-3-5	5/5/111	79	79	1804	406	1961	45	1832	67	134	133
nld-4-5	5/5/?	>2000	>2000	>2000	?	>2000	?	>2000	?	?	?
nld-7-3	3/3/7	96	95	5.8	9.8	9.1	7.7	5.8	11	0.37	0.16
nld-8-3	3/3/8	457	456	4	29	21	17	4	25	4.3	2.4
nld-9-3	3/3/7	1785	1777	39	43	121	70	40	45	0.34	0.29
nld-10-3	3/3/8	>2000	>2000	26	148	370	308	25	148	8.1	8.1
nql-5-4	5/5/2	109	102	0.1	1.3	0.12	1.3	0.1	0.39	0.27	0.39
nql-10-2	10/10/2	250	225	0.15	3.1	0.29	3.2	0.2	0.98	0.67	0.99
nql-10-4	10/10/2	>2000	>2000	0.33	3.2	0.61	3.3	0.34	0.92	0.62	0.83
nql-15-2	15/15/2	>2000	>2000	0.36	5.8	0.65	5.7	0.33	3.1	1.3	1.9
p3p-special	5/5/24	0.41	0.46	0.23	23	0.69	31	0.24	6.4	8.2	9
PlateForme2d-easy	6/6/0	<0.1	<0.1	1.1	0.12	1.4	0.12	0.99	<0.1	<0.1	<0.1
r-5	5/5/1	1.6	1.6	0.43	<0.1	0.49	<0.1	0.37	<0.1	<0.1	<0.1
r-6	6/6/1	>2000	>2000	0.96	<0.1	1.2	<0.1	0.98	<0.1	<0.1	<0.1
Rose	3/3/18	0.63	0.67	0.72	39	1.1	59	0.71	5	22	20
simple-nql-20-30	20/20/2	>2000	>2000	0.57	28	0.88	28	0.63	65	2.8	0.33
Takeuchi-Lu	4/4/14	<0.1	<0.1	0.22	7.2	0.23	9.7	0.17	0.87	4.4	3.1
Trinks-2	6/7/0	<0.1	<0.1	<0.1	<0.1	0.11	<0.1	<0.1	<0.1	<0.1	<0.1
Trinks-difficult	6/6/2	<0.1	<0.1	0.13	2.8	0.22	4	0.13	0.34	1.4	1.3
wilkinson20	1/1/21	<0.1	<0.1	<0.1	1	<0.1	0.98	<0.1	0.12	0.49	0.39
wilkinsonxy	2/2/25	<0.1	<0.1	<0.1	8	<0.1	7.9	<0.1	2.8	2.6	2.6

Figure 1: Benchmarks

The second family of systems which causes difficulties to RootFinding[Isolate] are the nld-*d*-*n* (for non leading linear) defined by n equations of the form $x_1 + \cdots + x_{i-1} + x_i^d + x_{i+1} + \cdots + x_n - 1 = 0$ for $1 \leq i \leq n$. On those systems the computations performed by Triangularize tend to split into many branches, even though the equiprojectable decomposition consists of a few components (generally 2). For System nld-9-3, the command Triangularize (used without normalization option) produces 16 components where the largest coefficient has size 20 digits. The command EquiprojectableDecomposition (which requires the use normalized regular chains) produces 3 components for nld-9-3, where most coefficients have more than 1,000 digits. Since nld-9-3 has 729 complex solutions, this suggests that the univariate polynomial in the rational univariate representation has degree 729 and coefficients with size at least 1,000 digits. This makes it difficult to isolate the real roots of such polynomial. Therefore, the nld-*d*-*n* examples show that splitting can help solving some problems.

5 Conclusion

We presented a generalization of the Vincent-Collins-Akritas Algorithm for zerodimensional squarefree regular chains, and its implementation in Maple. Each box isolating a root can be refined arbitrarily after being computed. This allows manipulating algebraic numbers (encoded by a isolation box and a regular chain) very much like in [25]. Many improvements of the algorithm RealRootIsolate are possible and should be investigated. Among them, we believe that writing a C library to perform the isolation would improve a lot the timings.

Yet for some non-equiprojectable varieties, our algorithm and its MAPLE implementation show favorable performances.

References and Notes

[1] Alkiviadis G. Akritas. There is no Uspensky's method. In *proceedings of ISSAC 1986*, 1986.

[2] Alkiviadis G. Akritas, Adam W. Strzeboński, and Panagiotis S. Vigklas. Implementations of a new theorem for computing bounds for positive roots of polynomials. *Computing*, 78(4):355–367, 2006.

[3] Alkiviadis G. Akritas and Panagiotis S. Vigklas. A Comparison of Various Methods for Computing Bounds for Positive Roots of Polynomials. *Journal of Universal Computer Science*, 13(4):455–467, 2007.

[4] Philippe Aubry and Marc Moreno Maza. Triangular sets for solving polynomial systems: A comparative implementation of four methods. *J. Symb. Comp.*, 28(1-2):125–154, 1999.

[5] Eberhard Becker, Teo Mora, Maria G. Marinari, and Carlo Traverso. The shape of the shape lemma. In *Proc. of the international symposium on Symbolic and algebraic computation*, pages 129–133, New York, NY, USA, 1994. ACM Press.

[6] Martine Ceberio and Vladik Kreinovich. Greedy algorithms for optimizing multivariate horner schemes. *SIGSAM Bull.*, 38(1):8–15, 2004.

[7] Jin-San Cheng, Xiao-Shan Gao, and Chee-Keng Yap. Complete numerical isolation of real roots in zero-dimensional triangular systems. *Journal of Symbolic Computation*, 44(7):768 – 785, 2009.

[8] George E. Collins and Alkiviadis G. Akritas. Polynomial real root isolation using Descartes'rule of signs. In *proceedings of ISSAC'76*, pages 272–275, Yorktown Heights NY, 1976.

[9] George E. Collins, Jeremy R. Johnson, and Werner Krandick. Interval arithmetic in cylindrical algebraic decomposition. *J. Symb. Comput.*, 34(2):145–157, 2002.

[10] Xavier Dahan, Marc Moreno Maza, Éric Schost, Wenyuan Wu, and Yuzhen Xie. Lifting techniques for triangular decompositions. In *ISSAC'05*, pages 108–115. ACM Press, 2005.

[11] Xavier Dahan and Éric Schost. Sharp estimates for triangular sets. In *ISSAC 04*, pages 103–110. ACM, 2004.

[12] Jean Della Dora, Claire Dicrescenzo, and Dominique Duval. About a new method for computing in algebraic number fields. In *Proc. EUROCAL 85 Vol. 2*, volume 204 of *Lect. Notes in Comp. Sci.*, pages 289–290. Springer-Verlag, 1985.

[13] René Descartes. *Géométrie*. 1636.

[14] Arno Eigenwillig, Lutz Kettner, Werner Krandick, Kurt Mehlhorn, Susanne Schmitt, and Nicola Wolpert. A descartes algorithm for polynomials with bit-stream coefficients. In *CASC, volume 3718 of LNCS*, pages 138–149. Springer, 2005.

[15] Jean-Charles Faugère. A new efficient algorithm to compute Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139:61–88, 1999.

[16] Jeremy R. Johnson and Werner Krandick. Polynomial real root isolation using approximate arithmetic. In *Proceedings of the 1997 international symposium on Symbolic and algebraic computation*, pages 225–232. ACM, 1997.

[17] François Lemaire, Marc Moreno Maza, and Yuzhen Xie. The `RegularChains` library. In Ilias S. Kotsireas, editor, Maple Conference 2005, pages 355–368, 2005.

[18] Xin Li, Marc Moreno Maza, and Wei Pan. Computations modulo regular chains. In *proceedings of ISSAC'09*, pages 239–246, New York, NY, USA, 2009. ACM Press.

[19] Xin Li, Marc Moreno Maza, Raqeeb Rasheed, and Éric Schost. The MODPN library: Bringing fast polynomial arithmetic into MAPLE. In *MICA'08*, 2008.

[20] Zhengyi Lu, Bi He, Yong Luo, and Lu Pan. An algorithm of real root isolation for polynomial systems. In Dongming Wang and Lihong Zhi, editors, *Proceedings of Symbolic Numeric Computation 2005*, pages 94–107, 2005.

[21] Scott McCallum. An improved projection operation for cylindrical algebraic decomposition. In B. F. Caviness and J. R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 242–268. Springer, 1988.

[22] Marc Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report TR 4/99, NAG Ltd, Oxford, UK, 1999. Presented at the MEGA-2000 Conference, Bath, England.

[23] Bernard Mourrain and Jean-Pascal Pavone. Subdivision methods for solving polynomial equations. Technical report, INRIA, August 2005. (number RR-5658).

[24] Bernard Mourrain, Fabrice Rouillier, and Marie-Françoise Roy. The Bernstein Basis and Real Root Isolation. *Combinatorial and Computational Geometry, MSRI Publications*, 52:459–478, 2005.

[25] Renaud Rioboo. Real algebraic closure of an ordered field, implementation in axiom. In *Proc. ISSAC'92*, pages 206–215. ISSAC, ACM Press, 1992.

[26] Renaud Rioboo. Towards faster real algebraic numbers. *J. Symb. Comput.*, 36(3-4):513–533, 2003.

[27] Fabrice Rouillier. Solving zero-dimensional systems through the rational univariate representation. *AAECC*, 9:433–461, 1999.

[28] Fabrice Rouillier and Nathalie Revol. The multiple precision floating-point interval library (MPFI) library. http://gforge.inria.fr/projects/mpfi/.

[29] Fabrice Rouillier and Paul Zimmermann. Efficient isolation of polynomial real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2003.

[30] Joseph Alfred Serret. *Cours d'Algèbre Supérieure*. Gauthier–Villars, Paris, 4 edition, 1877.

[31] James Victor Uspensky. *Theory of Equations*. McGraw – Hill Co., New–York, 1948.

[32] Alexandre Joseph Hidulphe Vincent. Sur la résolution des équations numériques. *Journal de Mathématiques Pures et Appliquées*, 1:341–372, 1836.

[33] Joachim von zur Gathen and Jürgen Gerhard. Fast algorithms for taylor shifts and certain difference equations. In *ISSAC*, pages 40–47, 1997.

[34] Bican Xia and Lu Yang. An algorithm for isolating the real solutions of semi-algebraic systems. *J. Symb. Comput.*, 34(5):461–477, 2002.

[35] Bican Xia and Ting Zhang. Real solution isolation using interval arithmetic. *Comput. Math. Appl.*, 52(6-7):853–860, 2006.

A Practical Implementation of a Modular Algorithm for Ore Polynomial Matrices

Howard Cheng[1*] and George Labahn[2]

[1] Department of Mathematics and Computer Science
University of Lethbridge, Lethbridge, Canada
howard.cheng@uleth.ca

[2] Symbolic Computation Group
David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, Canada
glabahn@uwaterloo.ca

.....

Abstract

We briefly review a modular algorithm to perform row reduction of a matrix of Ore polynomials with coefficients in $\mathbb{Z}[t]$, and describe a practical implementation in Maple that improves over previous modular and fraction-free versions. The algorithm can be used for finding the rank, left nullspace, and the Popov form of such matrices.

1 Introduction

Ore domains provide a general setting for describing the arithmetic of linear differential, difference, and q-difference operators. Systems of differential, difference and q-difference equations can then be defined via matrices of Ore operators (polynomials) evaluated at unknown functions. One can then make use of matrix constructions to investigate such systems. For example, performing row reduction on a matrix of Ore polynomials to simpler forms allows one to determine its rank and left nullspace which give the minimum number of equations needed to represent the system of equations [1]. When a transformation is invertible, then the normal form gives the matrix representing an equivalent system with a minimum number of equations. When the leading coefficient is triangular (as in the weak Popov form), then the normal form allows one to rewrite high-order operators (e.g. derivatives) in terms of lower ones [3]. These transformations can also be applied to the computation of greatest common right divisors (GCRDs) and least common left multiples (LCLMs) [2, 7, 8, 9], which represents the intersection and the union of the solution spaces of systems of equations.

The FFreduce algorithm [2] is a procedure for row reducing a matrix of Ore operators which performs row operations in a fraction-free way to reduce to simpler form while still controlling coefficient growth. This algorithm computes the rank and left nullspace of these matrices, and can be used to compute the row-reduced and weak Popov forms of shift polynomial matrices [2], as well as the Popov form of general Ore polynomial matrices [4]. It can also be used to compute a greatest common right divisor (GCRD) and a least common left multiple (LCLM) of such matrices. Besides their general use with systems of equations, LCLMs are also used in nonlinear control theory in order to define the notion of transfer function in some cases [6].

*Correspondence to: University of Lethbridge, 4401 University Drive, Lethbridge, Alberta, T1K 3M4, Canada.

A modular version of the FFreduce algorithm was developed by the authors to reduce the computational complexity [3]. In the modular algorithm, it was observed that the evaluation reduction $\mathbb{Z}_p[t][Z;\sigma,\delta] \to \mathbb{Z}_p[Z;\sigma,\delta]$ is not generally an Ore ring homomorphism [9]. Instead of performing the row operations on the Ore polynomial matrices directly, the problem was converted to one involving a system of linear equations over $\mathbb{Z}_p$. Larger striped Krylov matrices over $\mathbb{Z}_p$ was constructed and row reductions were performed on these matrices. Each Krylov matrix was constructed dynamically—rows were added depending on which row is selected as the pivot in each step. This was needed to ensure that the correct image was computed during the reduction in the presence of potential unlucky homomorphisms, even though unlucky homomorphisms occur rarely in practice. Thus, the modular algorithm was a trade-off between not exploiting polynomial arithmetic (or equivalently, the structure of the matrix) and the improved efficiency of coefficient arithmetic in simpler domains.

One obstacle in obtaining further improvement was that the row operations to reduce the Krylov matrix have to be done one step at a time, because it is not possible to construct the entire Krylov matrix *a priori* or the wrong system of solutions may have been solved. As a result, the only linear algebra subroutines in the `LinearAlgebra:-Modular` package in Maple used to accelerate the computation were operations on individual rows instead of the entire matrix. The resulting implementation has to switch back and forth between high-level Maple code and low-level compiled linear algebra subroutines that are significantly faster. In practice, the resulting modular algorithm was only faster than the corresponding fraction-free algorithm for very large inputs.

In this work, we investigate the applicability of linear algebra subroutines on blocks of matrices to speed up the computation. Assuming that the first evaluation point is "lucky," the Krylov matrices for the remaining evaluation points can be constructed and the entire matrix can be reduced with a few calls to the appropriate linear algebra subroutines. This allows more sophisticated implementations of linear algebra subroutines to speed up the reduction process (e.g. [5]).

2 Notation and Definitions

The definitions given here are similar to those in our previous works [2, 3].

For any vector of integers (also called *multi-index*) $\vec{\omega} = (\omega_1, \ldots, \omega_p)$, we denote by $|\vec{\omega}| = \sum_{i=1}^{p} \omega_i$. The vector $\vec{e}_i$ denotes the i-th unit vector (of the appropriate dimension) such that $(e_i)_j = \delta_{ij}$; we also have $\vec{e} = (1, \ldots, 1)$ (of the appropriate dimension). We denote by $Z^{\vec{\omega}}$ the diagonal matrix having Z^{ω_i} on the diagonal.

Let k be any field and let $\sigma : k \to k$ be an injective endomorphism of k. Then, a *derivation* $\delta : k \to k$ with respect to σ is an endomorphism of the additive group of k satisfying

$$\delta(rs) = \sigma(r)\delta(s) + \delta(r)s$$

for all r, $s \in k$. In this paper, we will examine Ore polynomial rings with coefficients in $\mathbb{Z}[t]$. That is, the ring $\mathbb{Z}[t][Z;\sigma,\delta]$ with σ an automorphism, δ a derivation and with the multiplication rule

$$Z \cdot a = \sigma(a)Z + \delta(a)$$

for all $a \in \mathbb{Z}[t]$. When $\delta = 0$, we call the polynomials *shift polynomials*. For brevity, we will use $\mathbb{Z}[t][Z]$ when the specific choices of σ and δ are not important.

Let $\mathbb{Z}[t][Z]^{m\times n}$ be the ring of $m \times n$ Ore polynomial matrices over $\mathbb{Z}[t]$. We shall adapt the following conventions for the remainder of this paper. Let $\mathbf{F}(Z) \in \mathbb{Z}[t][Z]^{m\times n}$,

$N = \deg \mathbf{F}(Z)$, and write

$$\mathbf{F}(Z) = \sum_{j=0}^{N} F^{(j)} Z^j, \text{ with } F^{(j)} \in \mathbb{Z}[t]^{m \times n}.$$

We also write $c_j(\mathbf{F}(Z)) = F^{(j)}$ to denote the j-th coefficient matrix. The *row degree* of an Ore polynomial matrix $\mathbf{F}(Z)$ is $\vec{\nu} = \operatorname{rdeg} \mathbf{F}(Z)$ if the i-th row has degree ν_i. Some useful properties of matrices of Ore polynomials, such as linear independence and rank, can be found in [2].

The problem of row reduction of Ore polynomial matrices can be formalized as follows. An Ore polynomial vector $\mathbf{P}(Z) \in \mathbb{Z}[t][Z]^{1 \times m}$ is said to have *order** $\vec{\omega}$ with respect to $\mathbf{F}(Z)$ if

$$\mathbf{P}(Z) \cdot \mathbf{F}(Z) = \mathbf{R}(Z) \cdot Z^{\vec{\omega}} \tag{1}$$

for some *residual* $\mathbf{R}(Z)$. The set of all vectors of a particular order $\vec{\omega}$ forms a $\mathbb{Q}[t][Z]$-module. An *order basis* for this module, $\mathbf{M}(Z) \in \mathbb{Z}[t][Z]^{m \times m}$ of row degree $\vec{\mu}$, is a basis such that

1. every row, $\mathbf{M}(Z)_{i,*}$, has order $\vec{\omega}$ for all $1 \le i \le m$;
2. the rows of $\mathbf{M}(Z)$ form a basis of the module of all vectors of order $\vec{\omega}$. That is, every $\mathbf{P}(Z) \in \mathbb{Q}[t][Z]^{1 \times m}$ of order $\vec{\omega}$ can be written as $\mathbf{P}(Z) = \mathbf{Q}(Z) \cdot \mathbf{M}(Z)$ for some $\mathbf{Q}(Z) \in \mathbb{Q}[t][Z]^{1 \times m}$;
3. the leading column coefficient is normalized. That is, there exists a nonzero $d \in \mathbb{Z}[t]$ such that
$$\mathbf{M}(Z) = d \cdot Z^{\vec{\mu}} + \mathbf{L}(Z)$$
where $\deg \mathbf{L}(Z)_{k,l} \le \mu_l - 1$.

An order basis represents all row operations to eliminate a specified number of low-order coefficients. An order basis of a particular order and degree, if it exists, is unique up to a constant multiple [2, Theorem 4.4]. When $\vec{\omega} = (mN+1) \cdot \vec{e}$ and $\mathbf{R}(Z)$ is the corresponding residual, the rows in $\mathbf{M}(Z)$ corresponding to the zero rows in $\mathbf{R}(Z)$ give a basis for the left nullspace of $\mathbf{F}(Z)$. However, it is not known *a priori* the row degree $\vec{\mu}$ of the order basis. A row-reduced form and weak Popov form, together with the unimodular transformation matrix, can be extracted from $\mathbf{M}(Z)$ and $\mathbf{R}(Z)$ if $\mathbf{F}(Z)$ is a matrix of shift polynomials [2]. In the general case of matrices of Ore polynomials, the computation of the Popov form can be formulated as a left nullspace computation and can be extracted from the result of an order basis computation [3].

3 The Modular Algorithm

A modular algorithm was given in [3] to compute the order basis and the residual. The fraction-free algorithm [2] can be reduced easily from $\mathbb{Z}[t][Z]$ to $\mathbb{Z}_p[t][Z]$ using Chinese remaindering. The usual issue of normalization of the image, detecting unlucky homomorphisms, and termination can be dealt with as described in [3]. It should be noted that the

*Orders in this paper will be with respect to $\mathbf{F}(Z)$ and it will not be explicitly stated for the remainder of the paper.

algorithm is output-sensitive in that the number of primes used is determined by the output size, and there is no need to verify the result (e.g. by trial division).

However, the reduction from $\mathbb{Z}_p[t][Z]$ to $\mathbb{Z}_p[Z]$ was not possible because the evaluation homomorphism $t \leftarrow \alpha$ is generally not an Ore ring homomorphism. Instead, we formulate the order basis problem as a system of linear equations over $\mathbb{Z}_p$ and perform Gaussian elimination on the coefficient matrix. The method we follow is similar to polynomial GCD computation by Gaussian elimination on the well-known Sylvester matrix [10]. It can also be considered an extension to the modular algorithm for Ore polynomial GCRD computation of Li and Nemes [9].

Given row degree $\vec{\mu}$ and order $\vec{\omega}$, the coefficients in the order basis $\mathbf{M}(Z)$ can be viewed as a solution to a linear system of equations over the coefficient ring. By equating the coefficients of like powers, each row of the order basis satisfies a system of equations of the form

$$\begin{array}{c} \\ \left[\begin{array}{c|ccc|c} \cdots & Z^0 & \cdots & Z^{\mu_k-1+\delta_{1,k}} & \cdots \\ & p_k^{(0)} & \cdots & p_k^{(\mu_k-1+\delta_{1,k})} & \end{array}\right] \end{array} \cdot \begin{array}{c} Z^0 \quad\quad\quad\quad Z^{\vec{\omega}-\vec{e}} \\ \left[\begin{array}{c|c|c} & \vdots & \\ \hline \cdots & Z^0 \cdot F_{k,\cdot}(Z) & \cdots \\ & \vdots & \\ \cdots & Z^{\mu_k-1+\delta_{1,k}} \cdot F_{k,\cdot}(Z) & \cdots \\ \hline & \vdots & \end{array}\right] \end{array} = \mathbf{0}. \tag{2}$$

More formally, for any $\mathbf{P}(Z) \in \mathbb{Q}[t][Z]^{m\times n}$ we define

$$\mathbf{P}_{\vec{v}} = \left[P_{*,1}^{(0)} \quad \cdots \quad P_{*,1}^{(v_1)} | \cdots | P_{*,n}^{(0)} \quad \cdots \quad P_{*,n}^{(v_n)}\right]. \tag{3}$$

We also define (recall that $\vec{\omega} = \sigma \cdot \vec{e}$)

$$K(\vec{\mu},\vec{\omega}) = \begin{bmatrix} c_0(\ \vdots\ \mathbf{F}(Z)_{1,*}) & \cdots & c_{\sigma-1}(\ \vdots\ \mathbf{F}(Z)_{1,*}) \\ c_0(Z^{\mu_1}\cdot \vdots\ \mathbf{F}(Z)_{1,*}) & \cdots & c_{\sigma-1}(Z^{\mu_1}\cdot \vdots\ \mathbf{F}(Z)_{1,*}) \\ c_0(\ \vdots\ \mathbf{F}(Z)_{m,*}) & \cdots & c_{\sigma-1}(\ \vdots\ \mathbf{F}(Z)_{m,*}) \\ c_0(Z^{\mu_m}\cdot \vdots\ \mathbf{F}(Z)_{m,*}) & \cdots & c_{\sigma-1}(Z^{\mu_m}\cdot \vdots\ \mathbf{F}(Z)_{m,*}) \end{bmatrix}. \tag{4}$$

Then the i-th row of the order basis satisfies

$$(\mathbf{M}_{i,*})_{\vec{\mu}-\vec{e}+\vec{e}_i} \cdot K(\vec{\mu}-\vec{e}+\vec{e}_i,\vec{\omega}) = \mathbf{0}. \tag{5}$$

The matrix $K(\vec{\mu},\vec{\omega})$ has dimensions $|\vec{\mu}+\vec{e}| \times |\vec{\omega}|$, and is called a *striped Krylov matrix* (with m stripes). This is a generalization of the well-known Sylvester matrix when $m = 2$ and $n = 1$. We also define $K^*(\vec{\mu},\vec{\omega})$ to be the matrix $K(\vec{\mu},\vec{\omega})$ with linearly dependent columns removed.

Example 3.1 *Let* $\vec{\mu} = (2,2)$, $\vec{\omega} = (3,3)$, *and*

$$\mathbf{F}(Z) = \begin{bmatrix} 2Z^2+3tZ+6t^2 & Z^2-Z+2 \\ (t-1)Z+3t^3 & 3tZ+t \end{bmatrix} \in \mathbb{Z}[t][Z;\sigma,\delta]^{2\times 2}, \tag{6}$$

with $\sigma(a(t)) = a(t)$ *and* $\delta(a(t)) = \frac{d}{dt}a(t)$. *Then*

$$K(\vec{\mu},\vec{\omega}) = \left[\begin{array}{cc|cc|cc} 6t^2 & 2 & 3t & -1 & 2 & 1 \\ 12t & 0 & 6t^2+3 & 2 & 3t & -1 \\ 12 & 0 & 24t & 0 & 6t^2+6 & 2 \\ \hline 3t^3 & t & t-1 & 3t & 0 & 0 \\ 9t^2 & 1 & 3t^3+1 & t+3 & t-1 & 3t \\ 18t & 0 & 18t^2 & 2 & 3t^3+2 & t+6 \end{array}\right]. \tag{7}$$

One way to obtain an order basis of degree $\vec{\mu}$ and order $\vec{\omega}$ is to perform Gaussian elimination on $K(\vec{\mu},\vec{\omega})$ so that the first $\vec{\omega}$ columns are eliminated. The rows in the submatrix $K(\vec{\mu}-\vec{e},\vec{\omega})$ are used for pivots in the elimination process, and the remaining rows give the residual $\mathbf{R}(Z)$. The order basis can be recovered from the transformation matrix corresponding to these rows.

Example 3.2 *Continuing from Example 3.1, we perform Gaussian elimination on* $K((2,2),(3,3))$ *using the first two rows of each stripe as pivots. After removing some common factors in each row to reduce the results, the resulting matrix is*

$$\left[\begin{array}{cccccc} 6\,t^2 & 2 & 3\,t & -1 & 2 & 1 \\ 0 & -4 & 6\,t^3-3\,t & 2\,t+2 & 3\,t^2-4 & -t-2 \\ 0 & 0 & 0 & 0 & -252\,t^5+270\,t^4-234\,t^3-22\,t^2-16\,t+16 & 882\,t^4-104\,t^2-56\,t-10 \\ 0 & 0 & -3\,t^2+2\,t-2 & 7\,t & -2\,t & -t \\ 0 & 0 & 0 & 21\,t^2-14 & 9\,t^4-12\,t^3+10\,t^2-8\,t+8 & -21\,t^3+11\,t^2-14\,t+2 \\ 0 & 0 & 0 & 0 & -126\,t^6+135\,t^5-180\,t^4-11\,t^3+118\,t^2-20\,t & 441\,t^5-52\,t^3-28\,t^2-103\,t \end{array}\right] \tag{8}$$

with the corresponding transformation matrix

$$\left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ -2 & t & 0 & 0 & 0 & 0 \\ -6\,t^2+4 & -126\,t^4+6\,t^3-4\,t^2+4\,t+14 & 21\,t^3-14\,t & -252\,t^2+24\,t+34 & 252\,t^3-12\,t^2-34\,t-8 & 0 \\ -t & 0 & 0 & 2 & 0 & 0 \\ -3\,t^2+2 & 3\,t^3-2\,t^2+2\,t & 0 & 12\,t-4 & -6\,t^2+4\,t-4 & 0 \\ -3\,t^3+2\,t & -63\,t^5+3\,t^4-2\,t^3+2\,t^2+21\,t & 0 & -126\,t^3+12\,t^2+59\,t & 126\,t^4-6\,t^3-59\,t^2-4\,t & 21\,t^3-14\,t \end{array}\right]. \tag{9}$$

The order basis $\mathbf{M}(Z)$ *of degree* $\vec{\mu}=(2,2)$ *and order* $\vec{\omega}=(3,3)$ *can be easily extracted. The rows of* $\mathbf{M}(Z)$ *are:*

$$\left[(21t^3-14t)Z^2+(-126t^4+6t^3-4t^2+4t+14)Z-6t^2+4 \quad (252t^3-12t^2-34t-8)Z-252t^2+24t+34\right]$$

and

$$\left[(-63t^4+3t^3-2t^2+2t+21)Z-3t^2+2 \quad (21t^3-14t)Z^2+(126t^3-6t^2-59t-4)Z-126t^2+12t+59\right].$$

Unfortunately, the row degree $\vec{\mu}$ of the order basis $\mathbf{M}(Z)$ of order $\vec{\omega}$ is not known *a priori*. In practice, one starts with $\vec{\mu}_0 = \vec{0}$ and performs elimination on $K(\vec{\mu}_0,\vec{\omega})$. For any $i \geq 0$, $\vec{\mu}_{i+1}$ is determined by the pivoting needed to reduce $K(\vec{\mu}_i,\vec{\omega})$ by one more column. Thus, each step in the algorithm involves performing Gaussian elimination of one column followed by adding one row to the matrix. Unlucky homomorphisms occur when the determinant of $K^*(\vec{\mu},\vec{\omega})$ vanishes under the evaluation $t \leftarrow \alpha$. In such case, the pivoting that occurs during the elimination is different. Unlucky homomorphisms can be detected by comparing the different row degrees of the final order basis computed under each evaluation homomorphism.

The `LinearAlgebra:-Modular` package in Maple was used to perform efficient computations over $\mathbb{Z}_p$. The use of Gaussian elimination for solving the system of linear equations instead of working on the Ore polynomial matrices directly means that the modular algorithm is no longer exploiting the structure present in the Krylov matrix. On the other

hand, coefficient arithmetic over $\mathbb{Z}[t]$ can be replaced by simpler coefficient arithmetic over $\mathbb{Z}_p$. For larger problems, the gain in simpler coefficient arithmetic more than offsets the loss in efficiency by not exploiting the structure. The algorithm outperforms the fraction-free algorithm [2] for very large problems even though the fraction-free algorithm exploits the structure of the Krylov matrix. However, the modular algorithm is not competitive for small input [3].

4 Improved Implementation

The implementation of the modular algorithm described in [3] has two drawbacks. First, the interleaving between matrix construction and row elimination means that routines such as Gaussian elimination (on an entire matrix) or block matrix multiplication cannot be applied to speed up the computation further. The implementation would have to switch between high-level Maple code and the faster, low-level compiled code in the `LinearAlgebra:-Modular` package. Second, the extra work and bookkeeping required for incremental matrix construction reduce the advantage of the modular algorithm. We would like to make use of low-level compiled linear algebra routines as much as possible without switching to Maple code.

In order to improve the modular algorithm, we note the unpredictability of the final row degree is mostly due to the presence of unlucky homomorphisms, but they occur rarely in practice. Therefore, the incremental elimination algorithm given previously [3] is used on one evaluation point in $\mathbb{Z}_p$. Assuming that the evaluation point (and the prime p) is not unlucky, the order basis computed has the correct degree $\vec{\mu}$. If $\vec{\mu}$ turns out to be incorrect, it will be detected when combined with the results from other primes. In that case, we perform extra computations in $\mathbb{Z}_p$ that are wasted. However, it does not occur often in practice.

When the correct degree $\vec{\mu}$ of the order basis is known (as assumed), it is relatively straightforward to compute the order basis and the residual:

1. construct $\mathbf{A} = [K(\vec{\mu}, \vec{\omega} + (N+1)\cdot\vec{e}) \mid \mathbf{I}]$;
2. perform Gaussian elimination on $\mathbf{A}$ to compute a reduced row echelon form to eliminate the first $|\vec{\omega}|$ columns, using only rows in $K(\vec{\mu} - \vec{e}, \vec{\omega})$ as pivots;
3. record the linearly dependent columns J as well as $d = \det K^*(\vec{\mu} - \vec{e}, \vec{\omega})$ which is (up to sign) the product of the pivots used;
4. construct $\mathbf{B}$ from $-\mathbf{A}_{*,J}$ after removing the pivot rows and inserting the $m \times m$ identity matrix into the columns corresponding to those rows.
5. compute $\mathbf{C} = (-1)^{\sum_{i=2}^{m} \mu_i} \cdot d \cdot \mathbf{B} \cdot \mathbf{A}$;
6. if $\mathbf{C}$ is not zero in the first $|\vec{\omega}|$ columns, then the homomorphism is unlucky. Otherwise, extract $\mathbf{R}(Z)$ from the left part and $\mathbf{M}(Z)$ from the right part of $\mathbf{C}$.

The Gaussian elimination in Step 2 can be performed, for example, by calling the `RowReduce` routine in the `LinearAlgebra:-Modular` package of Maple on the entire matrix $\mathbf{A}$. The matrix multiplication in Step 5 can be performed by the `Multiply` routine. As a result, the new implementation can fully take advantage of good low-level implementation of block Gaussian elimination and multiplication (e.g. [5]). Since there is no need to perform incremental matrix construction, both memory management and bookkeeping are reduced. In

addition, the control of the program can stay inside the low-level `LinearAlgebra:-Modular` subroutines instead of switching back and forth between them and Maple code.

Example 4.1 *We apply this method to Example 3.1. We perform our calculations in* $\mathbb{Z}_{31}$ *and perform the evaluation* $t \leftarrow 7$*. To conserve space, we only show* $\mathbf{A}' = [K(\vec{\mu}, \vec{\omega}) \mid \mathbf{I}]$ *and compute only* $\mathbf{M}(Z)$*. Initially,*

$$\mathbf{A}' = \begin{bmatrix} 15 & 2 & 21 & 30 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 22 & 0 & 18 & 2 & 21 & 30 & 0 & 1 & 0 & 0 & 0 & 0 \\ 12 & 0 & 13 & 0 & 21 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 6 & 7 & 6 & 21 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 7 & 1 & 7 & 10 & 6 & 21 & 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 14 & 2 & 8 & 13 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{10}$$

Performing Gaussian elimination on rows 1, 2, 4, and 5, we obtain:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 14 & 14 & 29 & 5 & 0 & 27 & 1 & 0 \\ 0 & 1 & 0 & 0 & 25 & 9 & 11 & 1 & 0 & 24 & 28 & 0 \\ 12 & 0 & 13 & 0 & 21 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 14 & 25 & 0 & 23 & 0 & 6 & 20 & 0 \\ 0 & 0 & 0 & 1 & 25 & 8 & 22 & 2 & 0 & 21 & 26 & 0 \\ 2 & 0 & 14 & 2 & 8 & 13 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

Here, $J = \{1, 2, 3, 4\}$ *and* $d = 26$*. Thus,*

$$\begin{aligned} \mathbf{C} &= 26 \cdot \begin{bmatrix} -12 & 0 & \mathbf{1} & -13 & 0 & \mathbf{0} \\ -2 & 0 & \mathbf{0} & -14 & -2 & \mathbf{1} \end{bmatrix} \cdot \mathbf{A}' \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 6 & 4 & 28 & 26 & 26 & 27 & 0 \\ 0 & 0 & 0 & 0 & 28 & 14 & 14 & 6 & 0 & 1 & 27 & 26 \end{bmatrix}, \end{aligned}$$

where the highlighted entries are the identity matrix inserted to form $\mathbf{B}$*. Therefore, the image of the order basis computed is*

$$\mathbf{M}(Z) = 5 \cdot \begin{bmatrix} 6Z^2 + 16Z + 20 & 11Z + 6 \\ 30Z + 8 & 6Z^2 + 11Z + 5 \end{bmatrix}. \tag{12}$$

One may easily verify that this is a scalar multiple of the image of the order basis computed in Example 3.2 under the evaluation $t \leftarrow 7$ *in* $\mathbb{Z}_{31}$*.*

The introduction of the scalar multiple is due to the removal of content in Example 3.2. The implementation given here in fact computes exactly the same result (including the scalar multiple) as the previous fraction-free and modular implementations for the order basis problem [2, 3].

5 Experimental Results

Experiments were performed on Ore polynomial matrices in differential case. The results of these experiments are shown in Tables 1 and 2. The application of block linear algebra routines reduces the running time of the modular algorithm in all cases. The improvement is more significant for smaller problems, where the original modular algorithm is not competitive against the fraction-free problems.

Table 1: Comparison of fraction-free, modular, and the new modular algorithm on random $m \times n$ matrices with $\deg_t = 1$ and integer coefficients having magnitude ≤ 5.

m, n	N	FFreduce (s)	Modular (s)	New Modular (s)	Improvement
2	1	0.023	0.115	0.069	40%
2	2	0.107	0.242	0.192	21%
2	4	1.689	2.984	2.301	23%
2	8	15.047	28.499	23.232	18%
2	16	278.883	279.041	232.877	17%
2	32	5447.542	4669.801	3992.689	15%
3	1	0.472	1.060	0.723	32%
3	2	3.808	6.268	5.416	20%
3	4	41.549	51.498	44.253	14%
3	8	667.682	599.466	521.571	13%
4	2	41.348	47.841	41.765	13%
4	4	663.707	554.060	487.000	12%
4	6	3850.143	2561.281	2303.989	10%
5	2	293.122	260.021	227.314	13%
5	4	6179.169	3845.945	3362.376	13%
8	1	1660.258	1088.609	998.659	8%
10	1	16179.879	8019.137	7524.240	6%

Table 2: Comparison of fraction-free, modular, and the new modular algorithm on random $m \times n$ matrices with $\deg_t = 2$ and integer coefficients having magnitude ≤ 5.

m, n	N	FFreduce (s)	Modular (s)	New Modular (s)	Improvement
2	2	0.470	1.647	1.378	16%
2	4	5.920	11.611	10.004	14%
2	8	86.214	128.528	109.911	14%
2	16	1237.410	1437.492	1300.804	10%
3	2	14.718	25.101	21.954	13%
3	4	216.214	267.295	238.497	11%
3	6	1157.705	1220.524	1114.106	9%
3	8	3933.234	3994.955	3735.837	6%
4	2	170.561	193.981	174.399	10%
4	4	2397.460	2270.272	2096.580	8%

For the larger problems, however, the improvement is less significant. For larger problems, the size of the coefficients in the output becomes larger as well. More time is spent on the other parts of the algorithm such as reconstruction by Chinese remaindering and memory management, and the amount of time spent on actual elimination is proportionally smaller. Since the new implementation improves mainly the elimination process, the improvement is less significant for larger problems. On the other hand, we see that the improved implementation given in this paper increases the advantage of the modular algorithm

over the fraction-free algorithm, and allows the modular algorithm to be used beneficially for smaller problems.

References and Notes

[1] B. Beckermann, H. Cheng, and G. Labahn. Fraction-free row reduction of matrices of skew polynomials. In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 8–15. ACM, 2002.

[2] B. Beckermann, H. Cheng, and G. Labahn. Fraction-free row reduction of matrices of Ore polynomials. *Journal of Symbolic Computation*, 41(5):513–543, 2006.

[3] H. Cheng and G. Labahn. Modular computation for matrices of Ore polynomials. In *Computer Algebra 2006: Latest Advances in Symbolic Algorithms*, pages 43–66, 2007.

[4] P. Davies, H. Cheng, and G. Labahn. Computing Popov form of general Ore polynomial matrices. In *Milestones in Computer Algebra (MICA) 2008*, pages 149–156, 2008.

[5] J.-G. Dumas, P. Giorgi, and C. Pernet. FFPACK: finite field linear algebra package. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, pages 119–126. ACM, 2004.

[6] M. Halas, U. Kotta, Z. Li, H. Wang, and C. Yuan. Submersive rational difference systems and formal accessibility. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, pages 175–182. ACM, 2009.

[7] Z. Li. *A Subresultant Theory for Linear Differential, Linear Difference and Ore Polynomials, with Applications*. PhD thesis, RISC-Linz, Johannes Kepler University, Linz, Austria, 1996.

[8] Z. Li. A subresultant theory for ore polynomials with applications. In *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, pages 132–139. ACM, 1998.

[9] Z. Li and I. Nemes. A modular algorithm for computing greatest common right divisors of ore polynomials. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 282–289. ACM, 1997.

[10] R. Loos. Generalized polynomial remainder sequences. In *Computer Algebra: Symbolic and Algebraic Computation*, pages 115–137. Springer-Verlag, 1982.

Ramanujan graphs of larger girth

Xavier Dahan[1] and Jean-Pierre Tillich[2*]

[1] Kyûshû university, Dep[t] of Mathematics, JAPAN
dahan@math.kyushu-u.ac.jp

[2] INRIA Rocquencourt, Projet SECRET, FRANCE
jean-pierre.tillich@inria.fr

Abstract

We generalize the celebrated construction of the graphs of Lubotzky, Phillips and Sarnak in their classical work "Ramanujan graphs" [4]. Our new approach consists in using octonion algebras rather than quaternions. The families obtained by this mean present not only the same spectacular spectral property that make them good expanders, but also show a higher girth, yielding a new record for regular graphs.

1 Introduction

In this extended abstract, we define Ramanujan graphs by introducing first the notion that made them famous, the *expanders*.

The concept of *expansion* in a graph has several but closely related meanings, and plays a significant role at several places of computer science. We refer to the praising Introduction's of a work of M. Klawe, recopied in Chapter 1 of Lubotzky's book [3]. The more recent and complete survey [2] on expanders is worth consulting for an overview on the broad range of applications of expanders: complexity theory, derandomization, design of optimal codes, networks theory and more. The following lines are taken from [1, Introduction], where the references for the results presented can be found.

Let $X = (V(X), E(X))$ be a connected regular undirected graph of size $n = |V(X)|$ and valency k. Let $F \subset V(X)$ be a subset of the the vertices, and ∂F be the *boundary*, consisting of the set of edges connecting F to its complement $V(X) - F$. The **expanding coefficient** of X is:

$$h(X) := \inf \left\{ \frac{|\partial F|}{\min\{|F|, |V(X) - F|\}}, \text{ s.t. } F \subset V(X), \text{ and } |F| > 0 \right\}. \tag{1}$$

For example, the complete graph on n vertices K_n has $h(K_n) = \inf\{\frac{k(n-k)}{\min\{k,n-k\}} \text{ s.t. } 0 < k < n\}$, that is always equal to $n - \lceil n/2 \rceil \simeq n/2$.

On the other extreme, the cyclic graph on n vertices C_n has a small expanding coefficient. By taking F a half-cycle in Equation (1) comes $h(C_n) \leq 2/(n/2) \simeq 4/n$, so $h(C_n) \xrightarrow{n\uparrow\infty} 0$.

Definition 1 *A family of regular graphs $(X_n)_{n\in\mathbb{N}}$ of increasing size $|X_n|$, and of fixed valency k is a* family of expanders *if the graphs are connected, and if there exists $\epsilon > 0$ such that $h(X_n) > \epsilon$ for all $n \geq 1$.*

Of course, what is expected is a high expansion coefficient for a rather small valency. While it is easy to prove existence of expanders, it has been much more difficult to efficiently construct ones. Margulis [5] did it first, its proof using quite involved mathematical tools. Then

*Correspondence to: X. Dahan. Kyushu university, Dep[t] of Mathematics, Moto-oka 744, Nishi-ku, 819-0395 Fukuoka. Tel/Fax: +81 (0)92 802 4482

came *Ramanujan graphs* (Margulis [6] and [4] independently), that achieve the expansion property by taking another door, the "spectral" one, as explained hereunder.

The adjacency matrix of an undirected graph X is symmetric, so its eigenvalues are *real*, denoted and ordered as follows: $\lambda_0(X) \geq \lambda_1(X) \geq \cdots \geq \lambda_{n-1}(X)$. If the graph is regular of valency k, then k is an eigenvalue and it is the largest one, so $\lambda_0 = k$. If the graph is connected, then $\lambda_0 > \lambda_1$, and reciprocally. By a result of Dodziuk and independently Alon & Milman, if X is a connected, k-regular finite graph:

$$\frac{1}{2}(k - \lambda_1(X)) \leq h(X) \leq \sqrt{2k(k - \lambda_1(X))}. \tag{2}$$

The difference $k - \lambda_1$ is called the **spectral gap**. Inequalities (2) provide a convenient way to prove expansion for the regular graphs. However, the spectral gap, and hence the expansion coefficients, can not hoped to be too large, as the following theorem, proved by Alon & Boppana, states:

$$\liminf_{n\to\infty} \lambda_1(X_n) \geq 2\sqrt{k-1}. \tag{3}$$

Ramanujan graphs should be graphs with the asymptotic largest spectral gap. Equations (2) and (3) suggests that they are optimal expanders (well, actually only asymptotically, since it is not the case else, as seen by Kahale, see [2, § 4.6]).

Definition 2 *A finite, k-regular, connected and undirected graph X is a* Ramanujan graph *if $\lambda_1(X) \leq 2\sqrt{k-1}$.*

Given a fixed valency k, it is proved that Ramanujan graphs exist if $k-1$ is a power of prime (Cf. [10, Figure 2] or [2, § 5.3] for surveys apart from the finite *upper-half plane graphs* of A. Terras *et al.*).

The **girth** of a graph is the length of its shortest closed path, without backtracking. Given a family $(X_n)_{n\in\mathbb{N}}$ of k-regular graphs of increasing size, the following upper bound, proved by combinatorial observations, is well-known (a special instance of the Moore bound):

$$\text{girth}(X_n) \leq (2 + o_n(1)) \log_{k-1} |X_n|, \text{ with } o_n(1) \xrightarrow{n\uparrow\infty} 0. \tag{4}$$

The family $(X_n)_{n\in\mathbb{N}}$ is said to have **large girth** if there exists a constant $\gamma > 0$ such that $\text{girth}(X_n) \geq \gamma \log_{k-1} |X_n|$.

It is a difficult task to find graphs with large girth. Using the probabilistic method, Erdős and Sachs proved the *existence* of graphs of large girth with a constant $\gamma = 1$. The graphs of [4], that are explicit, present as to today the largest constant γ, namely $\gamma \geq 4/3$ (and actually $\gamma = 4/3$, from the work of Biggs & Boshier). Our new families of Ramanujan graphs will also have a large girth with a constant $\gamma = 12/7$, yielding a new record on the girth for regular graphs.

2 Outline of the constructions

Our construction is similar to the one of [4], on slightly more complicated objects, since non-associative, the *octonions*, whereas it was quaternions in [4]. It is worth recalling first this previous construction.

Let $p \equiv 1 \bmod 4$ and $q > p$ two prime numbers. The 2x2 matrices over $\mathbb{F}_q$ are isomorphic to any quaternion algebra over $\mathbb{F}_q$ (indeed, they are all isomorphic, called *split quaternions*,

denoted $\mathbb{H}(\mathbb{F}_q)$). For example, if $\mathcal{Z}_q$ is the center of the group of units $\mathbb{H}(\mathbb{F}_q)^\times$, we have: $\mathbb{H}(\mathbb{F}_q)^\times/\mathcal{Z}_q \simeq PGL_2(q)$ $(\star)$. Let us consider now *integer* quaternions. We denote by $1, \mathsf{i}, \mathsf{j}, \mathsf{ij}$ the usual basis of quaternion algebras, and by $N(\alpha)$ the norm of a quaternion α. A theory of (left) gcd, with a (left) Bézout identity exists since Hurwitz for integer quaternions. Hence a factorization into primes makes sense. The following distinguished set of prime divisors permit to gain an *unique* factorization.

$$\mathcal{P}(p) := \{\alpha = a_0 + a_1\mathsf{i} + a_2\mathsf{j} + a_3\mathsf{ij} \in \mathbb{H}(\mathbb{Z}),\ |\ N(\alpha) = p\ ,\ a_0 > 0,\ \text{and}\ \alpha - 1 \in 2\mathbb{H}(\mathbb{Z})\}. \quad (5)$$

Let $\Lambda(p)$ be the set of *reduced words* with letters taken in $\mathcal{P}(p)$, reduced meaning that two successive letters are not conjugate (if an element $\alpha \in \mathcal{P}(p)$ then its conjugate $\overline{\alpha} = 2a_0 - \alpha$ also). By unique factorization, it is possible to define a structure of free group on $\Lambda(q)$. Then the reduction modulo q, $\tau_q : \mathbb{H}(\mathbb{Z}) \to \mathbb{H}(\mathbb{F}_q)$, sends $\Lambda(p)$ (in fact, embedded in $\mathbb{H}(\mathbb{Z})$ but not through the inclusion) onto $\mathbb{H}(\mathbb{F}_q)^\times/\mathcal{Z}_q$. Next, let $\mathscr{S}(p,q) = \tau_q\big(\mathcal{P}(p)\big)$ modulo $\mathcal{Z}_q$.

Thanks to Isomorphism $(\star)$, we define the graphs $\mathscr{Y}_{p,q}$ of [4] as the Cayley graphs $\mathscr{C}ay\big(PGL_2(q), \mathscr{S}(p,q)\big)$ if $\left(\frac{p}{q}\right) = -1$, and as the Cayley graphs $\mathscr{C}ay\big(PSL_2(q), \mathscr{S}(p,q)\big)$ else, when p is a square modulo q. Indeed, $\mathscr{S}(p,q)$ only generates $PSL_2(q)$ in the later case, and $\mathscr{S}(p,q)$ generates fully $PGL_2(q)$ in the former. This shows that the graphs $\mathscr{Y}_{p,q}$ are connected. Moreover they are bipartite when $\left(\frac{p}{q}\right) = -1$ and else, not. Since $|\mathscr{S}(p,q)| = p+1$, we have the following proposition.

Proposition 1 *$\mathscr{Y}_{p,q}$ is $p+1$-regular of cardinality $q(q^2-q)$ if $\left(\frac{p}{q}\right) = -1$, and of cardinality $\frac{1}{2}q(q^2-1)$ else.*

The proof of the estimates on the girth [4, Theorem 3.4] is very easy, and is similar for the new graphs the same proof is used.

Theorem 1 $\operatorname{girth}(\mathscr{Y}_{p,q}) \geq 4\log_p q - \log_p 4$ *if* $\left(\frac{p}{q}\right) = -1$, *else* $\operatorname{girth}(\mathscr{Y}_{p,q}) \geq 2\log_p q$.

This yields the constant $4/3$ mentioned in Introduction, since $4\log_p q \simeq 4/3\log_{|\mathscr{S}(p,q)|-1}|\mathscr{Y}_{p,q}|$.

The new construction based on octonions. In the same way, octonion algebras over $\mathbb{F}_q$ are all isomorphic and called *split octonions*. The set of invertible elements $\mathbb{O}(\mathbb{F}_q)^\times$ is no more a group (non-associative) but, this weaker structure is called a *loop*. While non-associative, octonions are *composition algebras*, (the norm is multiplicative) hence are *alternative* and they verify the *Moufang identities*: $a((bc)a) = (a(bc))a = a(bc)a$. The loop $\mathbb{O}(\mathbb{F}_q)^\times$ is a *Moufang loop*, and in analogy with $PSL_2(q)$, let M be the normal subloop of $\mathbb{O}(\mathbb{F}_q)^\times$ of elements of norm 1. We introduce the normal central subloops $\mathcal{Z}_{\mathbb{O}}$ and $\mathcal{Z}_M$ of $\mathbb{O}(\mathbb{F}_q)^\times$ and M respectively, so that we have a loop embedding $M/\mathcal{Z}_M \hookrightarrow \mathbb{O}(\mathbb{F}_q)^\times/\mathcal{Z}_{\mathbb{O}}$. Paige proved [8] that $M/\mathcal{Z}_M$ is a simple loop, and an index 2 normal subloop of $\mathbb{O}(\mathbb{F}_q)^\times/\mathcal{Z}_{\mathbb{O}}$ (in total analogy with PGL_2 and PSL_2).

Let $1, \mathsf{i}, \mathsf{j}, \mathsf{ij}, \mathsf{t}, \mathsf{it}, \mathsf{jt}, (\mathsf{ij})\mathsf{t}$ the be the usual basis of the octonions over $\mathbb{Q}$. We denote by $N(\alpha)$ the norm of an octonion α. As done with quaternions, we define the set:

$$\mathscr{P}(p) := \{\alpha = a_0 + a_1\mathsf{i} + a_2\mathsf{j} + a_3\mathsf{ij} + a_4\mathsf{t} + a_5\mathsf{it} + a_6\mathsf{jt} + a_7(\mathsf{ij})\mathsf{t} \in \mathbb{O}(\mathbb{Z}),$$
$$\text{such that } a_0 > 0,\ \text{and } N(\alpha) = p,\ \text{and } \alpha - 1 \in 2\mathbb{O}(\mathbb{Z})\} \quad (6)$$

Unique factorization for integral octonions is not trivial (even in a Coxeter *maximal arithmetic*, that permits the Euclidean division). Rehm very nicely proposed a "distortion"

of the Euclidean algorithm [9, Proposition 4.1] and reached unique factorization (for our purpose it is [9, Proposition 5.4, Theorem 5.7]). Unavoidably, we need to prescribe one bracketing:

Definition 3 *A* reduced word *of length ℓ over a set of octonions $\mathcal{A}$ is an element*

$$\Big(\cdots\big(\alpha_1\alpha_2)\alpha_3\big)\cdots\Big)\alpha_\ell,$$

such that $\alpha_i \in \mathcal{A}$ and $\alpha_i \neq \overline{\alpha_{i+1}}$ for $i = 1, \ldots, \ell - 1$.

Let $\mathscr{L}(p)$ the set of all reduced words of any length over $\mathscr{P}(p)$. It can be made a loop, in which two different words will always gives two distinct elements in $\mathbb{O}(\mathbb{Z})$ (kind of a "free loop", but not trivially included in $\mathbb{O}(\mathbb{Z})$). The reduction modulo q, $\tau_q : \mathbb{O}(\mathbb{Z}) \to \mathbb{O}(\mathbb{F}_q)$ permits to define a loop homomorphism $\mathscr{L}(p) \to \mathbb{O}(\mathbb{F}_q)^\times/\mathcal{Z}_{\mathbb{O}}$, that is onto. Let, $\mathscr{T}(p,q) := \tau_q\big(\mathscr{P}(p)\big)$ modulo $\mathcal{Z}_{\mathbb{O}}$.

Lemma 1 *If $\left(\frac{p}{q}\right) = -1$, then $\mathscr{T}(p,q)$ generates $\mathbb{O}(\mathbb{F}_q)^\times/\mathcal{Z}_{\mathbb{O}}$, and if $\left(\frac{p}{q}\right) = 1$, $\mathscr{T}(p,q)$ generates $M/\mathcal{Z}_M$.*

Cayley graphs are usually defined for groups, but it is possible to define them for loops as well (Cf. [7] for an exposition), under light restrictions.

Definition 4 *If $\left(\frac{p}{q}\right) = -1$ let $\mathscr{X}_{p,q}$ be the Cayley graphs $\mathscr{C}ay\big(\mathbb{O}(\mathbb{F}_q)^\times/\mathcal{Z}_{\mathbb{O}}, \mathscr{T}(p,q)\big)$, and if $\left(\frac{p}{q}\right) = 1$, the Cayley graph $\mathscr{C}ay\big(M/\mathcal{Z}_M, \mathscr{T}(p,q)\big)$.*

We have $|\mathscr{T}(p,q)| = p^3+1$, [9, Proposition 6.4] and $|\mathbb{O}(\mathbb{F}_q)^\times/\mathcal{Z}_{\mathbb{O}}| = q^7 - q^3$, [9, Lemma 3.2] This permits to prove:

Proposition 2 *The graphs $\mathscr{X}_{p,q}$ are p^3+1-regular, connected and undirected. They are bipartite of cardinal $q^7 - q^3$ if $\left(\frac{p}{q}\right) = -1$, and not bipartite of cardinal $\frac{1}{2}(q^7 - q^3)$ else.*

The construction being achieved, let us give the idea of the proofs of the estimates on the spectral property (Definition 2) and on the girth. A main difference with quaternions is that unlike Cayley graphs on groups, the $\mathscr{X}_{p,q}$ are (very probably) *not* vertex-transitive. This comes about in the (essential) fact that the closed paths (without backtracking) of given length at each vertex, are no more in one-one correspondence (Cf. [1, Corollary 1.4.7]). It can however easily bypassed (*e.g.*, little changes in [1, Lemma 4.4.2]), and the machinery of § 4.4 (until Prop. 4.4.3, then directly Remark 4.4.7) of [1] can be applied.

Another difference is the quadratic form arisen here, in 8 variables $f(x) = x_0^2 + 4q^2x_1^2 + \cdots + 4q^2x_7^2$, similar to the one in 4 variables of [4, Equation (1.3)] (we need the estimate of Ramanujan-Petersson on the number of integer solutions of $f(x) = p^m$). The theta function associated to f is a modular form of weight 4. In [4], it had weight 2, the Eichler's proof of the Ramanujan-Petersson conjecture was sufficient. Here it is required the Deligne's reduction of the Ramanujan-Petersson conjecture for modular forms of even weight to the Weil conjectures (that he himself happened to prove).

To prove the estimate on the girth of the non vertex-transitive graphs $\mathscr{X}_{p,q}$, we need to consider cycle paths at *each* nodes and not only at the "origin". This appears, as said above, to be not a problem and a the same estimates as in Theorem 1 (with the same proof) are obtained. The characteristics of the graphs being different, we have:

$$\operatorname{girth}(\mathscr{X}_{p,q}) \geq 4\log_p q = \frac{12}{7}\log_{p^3} q^7 \geq \frac{12}{7}\log_{|\mathscr{T}(p,q)|-1}|\mathscr{X}_{p,q}|,$$

when p is not a square modulo q. That is the result announced in Introduction.

References and Notes

[1] G Davidoff, P Sarnak, and A Valette. *Elementary number theory, group theory, and Ramanujan graphs*, volume 55 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 2003.

[2] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc. (N.S.)*, 43(4):439–561 (electronic), 2006.

[3] A. Lubotzky. *Discrete groups, expanding graphs and invariant measures*, volume 125 of *Progress in Mathematics*. Birkhäuser Verlag, Basel, 1994. With an appendix by Jonathan D. Rogawski.

[4] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[5] G. A. Margulis. Explicit constructions of expanders. *Problemy Peredači Informacii*, 9(4):71–80, 1973.

[6] G. A. Margulis. Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. *Problemy Peredachi Informatsii*, 24(1):51–60, 1988.

[7] E. Mwambene. Cayley graphs on left quasi-groups and groupoids representing k-generalised Petersen graphs. *Discrete Math.*, 309(8):2544–2547, 2009.

[8] Lowell J. Paige. A class of simple Moufang loops. *Proc. Amer. Math. Soc.*, 7:471–482, 1956.

[9] HP Rehm. Prime factorization of integral Cayley octaves. *Ann. Fac. Sci. Toulouse Math. (6)*, 2(2):271–289, 1993.

[10] J.-P. Tillich and G. Zémor. Optimal cycle codes constructed from Ramanujan graphs. *SIAM J. Discrete Math.*, 10(3):447–459, 1997.

Computing Popov Forms of Matrices over PBW Extensions*
(Extended Abstract)

Mark Giesbrecht[1], George Labahn[1], Yang Zhang[2]

[1] School of Computer Science,
The University of Waterloo, Waterloo, Ontario, N2L 3G1 Canada
E-mail:mwg@uwaterloo.ca; glabahn@uwaterloo.ca

[2] Department of Mathematics,
The University of Manitoba, Winnipeg, MB, R3T 2N2, Canada
E-mail: zhang39@cc.umanitoba.ca

Abstract

In this paper we define the Popov and weak Popov forms of matrices over Poincaré-Birkhoff-Witt (PBW) extensions, and exhibit effective algorithms to find them. As applications we give general methods to calculate the ranks of such matrices, and a method to transfer a system of differential equations into a first order equation.

1 Introduction

When seeking to determine left and right equivalence properties of a matrix over a domain, one often seeks to transform that matrix into a canonical form, such as the Hermite or Smith form. These forms capture all left and two-sided equivalence properties respectively, but suffer from substantial growth in the size of entries. For example, for polynomial matrices, their degrees are generally on the order of the size of the matrix, even when the degrees of the entries in the original matrix are small. In order to avoid this increase in degree, Popov (1972) introduced another normal form, which has been successfully applied to control theory (see for example, Kailath (1980)). This form has come to be known as the Popov normal form.

Recently, the Popov form has attracted considerable interest in the computer algebra field for its low degree and correspondingly efficient algorithms. For example, Villard (1996) and Beckermann et al. (1999) defined and discussed shifted-Popov forms. In Mulders and Storjohann (2003) they define the *weak Popov form*, which while not canonical, is easier to compute and elicits many of the matrix properties we are interested in.

In the non-commutative case, matrices over *Ore domains* (domains which satisfy the *Ore condition*, which essentially says that any two elements have a non-trivial left (or right) common multiple) have been considered at least since the 1930's (see Ore (1933), Jacobson (1943, 1996)). In Jacobson (1943) the Hermite and Smith forms of matrices over skew polynomial rings are considered. Recent treatments from a ring theoretic perspective can be found in Cohn (1995). In the computer algebra field, Abramov and Bronstein (2002) gave a method to calculate the ranks of matrices over skew polynomial rings $\mathsf{R}[x, x^{-1}; \sigma]$, and their method can be applied to Weyl algebras. Beckermann et al. (2006) use fraction-free methods to discuss weak Popov form of matrices over skew polynomial rings. Giesbrecht et al. (2005) discuss Popov forms of matrices over valuation domains. Davies et al. (2008) and Giesbrecht and Kim (2009) develop reductions to linear algebra over a (commutative) field to give polynomial-time algorithm for Popov and, respectively, Hermite forms of matrices over an Ore domain.

Matrices over multivariate polynomial rings and rings of differential operators have been extensively used in multidimensional linear systems since the mid-1970s (see, for example, Bose (1985), Park (2004), Youla and Pickel (1984), Zerz (2000), Hillebrand and Schmale (2001)), and also in other areas such as operator algebras; see for example, Finkel and Kamran (1997). This work motivates us not only to consider the univariate case, but also the case of multivariate non-commutative polynomial rings.

In this paper, we discuss matrices over noncommutative rings called Poincaré-Birkhoff-Witt (PBW) extensions, which includes most popular rings with derivations studied in computer algebra. At first, we consider the row spaces of matrices as left modules over base rings and define two kinds of reductions which are used to construct weak Popov forms and Popov forms respectively. The rank of matrices remains invariant under our reductions, and equals the number of nonzero rows of the (weak) Popov forms. Note that the ranks of matrices are independent of term orders. Therefore some term orders may produce ranks very quickly, while some may be considerably slower.

As an application, this gives a direct method to calculate the rank of matrices over Weyl algebras, as compared to the method given by Abramov and Bronstein (2002), which sets up a bijection between the Weyl algebra and a skew Laurent polynomial ring.

In this extended abstract, we first recall the definition of PBW extensions, and then outline how to define (weak) Popov forms of matrices over PBW extensions. We then present an algorithm to construct Popov forms. More results including algorithms and examples will appear in the forthcoming journal version.

2 Definitions and some results

The definition of a PBW extension was first given by Bell and Goodearl (1988). This lead to a unified treatment of many polynomial-type rings which are currently studied in associative ring theory and computer algebra.

Definition 2.1 *Let* R *and* E *be two associative rings with* $\mathsf{R} \subseteq \mathsf{E}$. E *is called a* (finite) PBW extension *of* R *if there exist* $x_1, x_2, \cdots, x_n \in \mathsf{E}$ *such that*

(a) monomials $x_1^{i_1} \cdots x_n^{i_n}$ *form a basis for* E *as a free left* R*-module, for* $i_1, \ldots, i_n \in \mathbb{N}$*;*

(b) $x_i r - r x_i \in \mathsf{R}$ *for each* $i = 1, \ldots, n$ *and any* $r \in \mathsf{R}$*;*

(c) $x_i x_j - x_j x_i \in \mathsf{R} + \mathsf{R}x_1 + \cdots + \mathsf{R}x_n$ *for all* $i, j = 1, \ldots, n$.

Let $\mathfrak{R} = \mathsf{R}\langle x_1, \ldots, x_n \rangle$ be a PBW extension of an Ore domain R. One can naturally define a term-ordering on $\mathfrak{R}$ which satisfies the usual multiplicative properties and respects degree; we refer to (Giesbrecht et al., 2002, Section 2) for details. Furthermore, we can extend the term ordering on $\mathfrak{R}$ to the left $\mathfrak{R}$-module $\mathfrak{R}^m$ and define leading monomials, leading coefficients and leading terms in the natural way. Throughout this paper assume that $\prec$ is a *term-over-position* admissible term order on $\mathfrak{R}^m$, i.e., the monomial term order has higher priority than the position in the vector; see Giesbrecht et al. (2009) for details.

We now define the notions of weak reduction and reduction as mechanisms for cancelling terms in vectors via the leading term (designated by $\mathrm{lt}(\cdot)$) of another vector.

Definition 2.2 *Given* $\vec{a}, \vec{b}, \vec{c} \in \mathfrak{R}^m$*, we say that*

(a) $\vec{a}$ weakly reduces *to* $\vec{c}$ *modulo* $\vec{b}$ *in one step if and only if* $\mathrm{lt}(\vec{b})$ *divides* $\mathrm{lt}(\vec{a})$ *and* $\vec{c} = \vec{a} - q_1\vec{b}$*, where* $q_1 \in \mathfrak{R}$ *is such that* $\mathrm{lt}(\vec{a}) = \mathrm{lt}(q_1\vec{b})$.

(b) $\vec{a}$ reduces *to* $\vec{c}$ *modulo* $\vec{b}$ *in one step if and only if* $\mathrm{lt}(\vec{b})$ *divides a term* $\vec{d}$ *that appears in* $\vec{a}$ *and* $\vec{c} = \vec{a} - q_2\vec{b}$*, where* $q_2 \in \mathfrak{R}$ *is such that* $\vec{d} = \mathrm{lt}(q_2\vec{b})$.

Thus, weak reduction uses the leading term of $\vec{b}$ to cancel the leading term of $\vec{a}$, whereas full reduction is much stronger, and uses the leading term of $\vec{b}$ to cancel *any* possible term in $\vec{a}$. The *weak division algorithm* (resp. the *division algorithm*) for $\vec{r} \in \mathfrak{R}^m$ by a set $\Gamma \subseteq \mathfrak{R}^m$ is defined correspondingly to reduce the least leading monomials (resp. all possible monomials) in $\vec{r}$ by the leading monomials of elements of Γ.

Definition 2.3 *A nonzero vector $\vec{a}$ in $\mathfrak{R}^m$ is called* (weakly) reduced *with respect to a set $S = \{\vec{s}_1, \ldots, \vec{s}_l\}$ of nonzero vectors in $\mathfrak{R}^m$ if no (leading) term that appears in $\vec{a}$ is divisible by any one of the* $\mathrm{lt}(\vec{s}_i),\ i = 1, \ldots, l$.

Furthermore, a set $S = \{\vec{s}_1, \ldots, \vec{s}_\ell\}$ of nonzero vectors is called (weakly) reduced *if each vector $\vec{s}_i$ (for $1 \le i \le \ell$) is (weakly) reduced with respect to $S \setminus \{\vec{s}_i\}$.*

We can now define the weak Popov and Popov forms as follows:

Definition 2.4 *Given a matrix $\mathfrak{R}^{m\times n}$ and a term-over-position admissible term order on $\mathfrak{R}^m$, let $\Gamma = \{\vec{r}_1, \ldots, \vec{r}_m\}$ be its set of its row vectors.*

(a) $\mathfrak{R}^{m\times n}$ is in weak Popov form *with respect to $\prec$ if Γ is a weakly reduced set.*

(b) $\mathfrak{R}^{m\times n}$ is in Popov form *with respect to $\prec$ if*

(i) Γ is a reduced set;

(ii) the leading coefficients of $\{\vec{r}_i\}$ are monic;

(iii) rows are in a descending chain with respect to $\prec$, that is, $\vec{r}_m \prec \cdots \prec \vec{r}_2 \prec \vec{r}_1$.

The weak reduction and reduction correspond to weak Popov form and Popov form, respectively. In this paper, we first describe two algorithms to present (weak) divisions, and then use them to construct (weak) Popov forms. Here we list one as follows:

Algorithm: `Popov form for` $\mathfrak{R}^{m\times n}$

Input: ‣ row vectors $\vec{r}_1, \ldots, \vec{r}_m$ of a matrix $A \in \mathfrak{R}^{m\times n}$;

Output: ‣ row vectors $\vec{p}_1, \ldots, \vec{p}_m$ of Popov form of $A \in \mathfrak{R}^{m\times n}$;

```
Initialization: p_1 := 0, ..., p_m := 0;  changes:=true;
While (changes) do
    changes := false;
    Swap rows so that they are in a descending chain with respect to ≺;
    For i from 2 to m do
        If r_i is reducible modulo r_1 then
            Using the division algorithm, let r_i equal the remainder of r_i by r_1;
            changes := true;
    end do;
end do;
Return: p_1 := r_1, ..., p_m := r_m;
Make leading coefficients 1 by multiplying by suitable elements of quotient field of R.
```

Theorem 2.5 *The Popov form algorithm terminates after a finite number of steps and produces a Popov form.*

The proof follows relatively easily from the fact that $\mathfrak{R}^m$ is noetherian, and the repeated reduction yields a descending chain of ideals which must be finite. Of course, the number of reduction steps is of great interest in the efficiency of our algorithms.

One of the applications of Popov forms is to provide the rank information efficiently. First we prove the following theorem which implies that the ranks of matrices are invariant under Popov form transformations.

Theorem 2.6 *For any matrix $A \in \mathfrak{R}^{m\times n}$ there exists a unimodular matrix U such that UA is in Popov form (similarly for weak Popov form).*

The transformation matrix U and the Popov form can be computed by our algorithms. As an application, we can find the rank of a matrix by counting the number of non-zero rows. Also, given a unimodular matrix $\mathfrak{R}^{n\times n}$ (one whose inverse is also in $\mathfrak{R}^{n\times n}$), one can find its inverse simply by noting that the Popov form of a unimodular matrix is the identity.

We also anticipate that the Popov form will be useful as an intermediate step to efficiently computing the Hermite and Smith/Jacobson forms in an effective manner.

References

S. Abramov and M. Bronstein. Linear algebra for skew-polynomial matrices. Technical Report 4420, INRIA, 2002. INRIA Rapport de Recherche.

B. Beckermann, G. Labahn, and G. Villard. Shifted normal forms of polynomial matrices. In *Proceedings of ISSAC'99*, pages 189–196. ACM Press, 1999.

B. Beckermann, H. Cheng, and G. Labahn. Fraction-free row reduction of matrices of Ore polynomials. *Journal of Symbolic Computation*, 41(5):513–543, 2006.

A. D. Bell and K. R. Goodearl. Uniform rank over differential operator rings and Poincaré-Birkhoff-Witt extension. *Pacific Journal of Mathematics*, 131(1):13–37, 1988.

N. K. Bose. *Multidimensional Systems Theory.* D. Reidel, 1985.

P. M. Cohn. *Skew fields.* Cambridge University Press, 1995.

P. Davies, H. Cheng, and G. Labahn. Computing Popov form of general Ore polynomial matrices. In *Milestones in Computer Algebra*, pages 149–156, 2008.

F. Finkel and N. Kamran. On the equivalence of matrix differential operators to Schrödinger form. *Nonlinear Mathematical Physis*, 4(3–4):278–286, 1997.

M. Giesbrecht and M. Kim. On computing the Hermite form of a matrix of differential polynomials. In *Proc. CASC'09*, 2009.

M. Giesbrecht, G. Reid, and Y. Zhang. Non-commutative Gröbner bases in Poincaré-Birkhoff-Witt extensions. In *Proc. Conference on Computer Algebra and Scientific Computation (CASC'02)*, pages 97–106, 2002.

M. Giesbrecht, G. Labahn, and Y. Zhang. Computing valuation Popov forms. In *Proc. CASA'2005*, Lecture Notes on Computer Science 3516, pages 619–626. Springer-Verlag, 2005.

M. Giesbrecht, G. Labahn, and Y. Zhang. Computing Popov forms of matrices over Ore domains. Preprint, 2009.

A. Hillebrand and W. Schmale. Towards an effective version of a theorem of Stafford. *J. of Symbolic Computation*, 32:699–716, 2001.

N. Jacobson. *The Theory of Rings.* American Math. Soc., New York, 1943.

N. Jacobson. *Finite-Dimensional Division Algebras over Fields.* Springer-Verlag, 1996.

T. Kailath. *Linear Systems.* Prentice Hall, 1980.

T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *Journal of Symbolic Computation*, 35:377–401, 2003.

O. Ore. Theory of non-commutative polynomials. *Annals of Mathematics*, 34(22):480–508, 1933.

H. Park. Symbolic computation and signal processing. *Journal of Symbolic Computaiton*, 37: 209–226, 2004.

V. M. Popov. Invariant description of linear, time-invariant controllable systems. *SIAM J. Control*, 10(2):252–264, 1972.

G. Villard. Computing Popov and Hermite forms of polynomial matrices. In *Proc. ISSAC'96*, pages 250–258. ACM Press, 1996.

D. C. Youla and P. F. Pickel. The Quillen-Suslin theorem and the structure of n-dimemsional elementary polynomial matrices. *IEEE Trans. Circuits Systems*, 31:513–518, 1984.

E. Zerz. *Topics in Multidimensional Linear Systems Theory.* Lecture Notes in Control and Information Sciences. Springer, 2000.

On the Implementation of Boolean Gröbner Bases

Shutaro Inoue[1] and Akira Nagai[2]

[1] Tokyo University of Science,
1-3, Kagurazaka, Shinjuku-ku, Tokyo, Japan
inoue@mi.kagu.tus.ac.jp

[2] NTT Information Sharing Platform Laboratories,
3-9-11, Midorimachi, Musashino, Tokyo, Japan
nagai.akira@lab.ntt.co.jp

Abstract

We show how we can make Boolean Gröbner bases computations feasible on standard computer algebra systems which have a routine to compute Gröbner bases in polynomial rings over the Galois field $\mathbb{GF}_2$. We also show that we can even compute a comprehensive Boolean Gröbner basis using only computations of Gröbner bases in a polynomial ring over $\mathbb{GF}_2$. Our implementation on the computer algebra system Risa/Asir achieves tremendous speedup comparing with previous implementations of Boolean Gröbner bases.

1 Introduction

A commutative ring $\mathbf{B}$ with an identity is called a *Boolean ring* if every element of which is idempotent. A residue class ring $\mathbf{B}[X_1, \ldots, X_n]/\langle X_1^2 - X_1, \ldots, X_n^2 - X_n\rangle$ with an ideal $\langle X_1^2 - X_1, \ldots, X_n^2 - X_n\rangle$ also becomes a Boolean ring, which is called a *Boolean polynomial ring* and denoted by $\mathbf{B}(X_1, \ldots, X_n)$. A Gröbner basis in a Boolean polynomial ring (called a *Boolean Gröbner basis*) is first introduced in [4, 5] and further developments are done in [2, 6, 7, 8, 10]. The original computation algorithm introduced in [4, 5] uses a special monomial reduction which is more complicated than a usual monomial reduction in a polynomial ring over a field. It is also directly applicable for the computations of comprehensive Boolean Gröbner bases. This algorithm is first implemented in Prolog as a free software([6]) for the computations of both Boolean Gröbner bases and comprehensive Boolean Gröbner bases for the case that $\mathbf{B}$ is a Boolean ring $\mathcal{P}_{FC}(S)$ that consists of all finite or co-finite subsets of S. (Here, S is a set of all strings of the computer language.)

It seems a very natural and easy way to implement them in a computer algebra system which has a facility to manipulate polynomials, however it is not very simple to implement computations of the above Boolean ring(including how to represent their data structures) in standard computer algebra systems. In [7], an alternative algorithm is introduced where we can obtain a Boolean Gröbner basis by only computing usual Gröbner bases in a polynomial ring over the Galois field $\mathbb{GF}_2$. Its implementation brought us a much faster program than [6]. Unfortunately, this algorithm is not applicable for the computations of comprehensive Boolean Gröbner bases, and no implementations had been done in any computer algebra system.

After a decade of the pioneering work of Boolean Gröbner bases, further developments are recently done in [2, 8, 10]. Based on these results, we have implemented a software [1] to compute both Boolean Gröbner bases and comprehensive Boolean Gröbner bases for the Boolean ring $\mathcal{P}_{FC}(S)$ in the computer algebra system Risa/Asir [3]. Our software achieves a tremendous speedup comparing with the previous one. It enables us to do our recent work [9] of a non-trivial application of Boolean Gröbner bases.

In this short paper, we describe how we can implement computations of both Boolean Gröbner bases and comprehensive Boolean Gröbner bases for the Boolean ring $\mathcal{P}_{FC}(S)$ in the computer algebra system Risa/Asir. We can also easily modify our method for any other computer algebra systems which have a routine to compute Gröbner bases in polynomial rings over the Galois field $\mathbb{GF}_2$.

The reader is referred to [2, 10] for the detailed descriptions of the properties of Boolean Gröbner bases which we use in this paper together with important definitions such as a Boolean closed polynomial, a reduced Boolean Gröbner basis and a stratified Boolean Gröbner basis.

2 Several key facts of Boolean Gröbner bases

Given a finite set F of Boolean polynomials in $\mathbf{B}(X_1, \ldots, X_n)$, let $\mathbf{B}'$ be its smallest Boolean subring that contains all coefficients of polynomials in F. Obviously, any Boolean Gröbner basis G of the ideal $\langle F \rangle$ in $\mathbf{B}'(X_1, \ldots, X_n)$ is also a Boolean Gröbner basis of the ideal $\langle F \rangle$ in $\mathbf{B}(X_1, \ldots, X_n)$. Note that $\mathbf{B}'$ is a finite Boolean ring, so it is isomorphic to a direct product $\mathbb{GF}_2^k$ of the Galois field $\mathbb{GF}_2$ for some natural number k.

In what follows, c_i denotes the i-th component of $c \in \mathbb{GF}_2^k$ for each $i = 1 \ldots, k$, f_i denotes the Boolean polynomial of $\mathbb{GF}_2(\bar{X})$ obtained from a Boolean polynomial f of $\mathbb{GF}_2^k(\bar{X})$ by replacing each coefficient c with c_i. $\bar{X}$ is an abbreviation of $X_1, \ldots, X_n$.

Computation of Boolean Gröbner bases in $\mathbb{GF}_2(\bar{X})$ is quite easy.

Theorem 1 *For a finite set $\{f_1, \ldots, f_l\}$ of Boolean polynomials in $\mathbb{GF}_2(\bar{X})$, let G be a (reduced) Gröbner basis of the ideal $\langle f_1, \ldots, f_l, X_1^2 - X_1, \ldots, X_n^2 - X_n \rangle$ in the polynomial ring $\mathbb{GF}_2[\bar{X}]$ over the field $\mathbb{GF}_2$ w.r.t. a term order.*
Then $G \setminus \{X_1^2 - X_1, \ldots, X_n^2 - X_n\}$ is a (reduced) Boolean Gröbner basis of the ideal $\langle f_1, \ldots, f_l \rangle$ in $\mathbb{GF}_2(\bar{X})$ w.r.t. the same term order.

The next theorem which is essentially a special instance of Theorem 2.3 of [11] plays an important role in the computation algorithm of Boolean Gröbner bases employed in [7].

Theorem 2 *In a Boolean polynomial ring $\mathbb{GF}_2^k(\bar{X})$, let G be a finite set of Boolean closed Boolean polynomials. Then, G is a (reduced) Boolean Gröbner basis of an ideal I in $\mathbb{GF}_2^k(\bar{X})$ if and only if $G_i = \{g_i | g \in G\} \setminus \{0\}$ is a (reduced) Gröbner bas is of the ideal $I_i = \{f_i | f \in I\}$ in $\mathbb{GF}_2(\bar{X})$ for each $i = 1, \ldots, k$.*

Example 1 *The following left constraint with unknown set variables X and Y is equivalent to the right system of equations of a Boolean polynomial ring $\mathbf{B}(X, Y)$, where $\mathbf{B}$ is a Boolean ring $\mathcal{P}_{FC}(S)$.*

$$\begin{cases} X \cup Y \subseteq \{s_1, s_2\} \\ s_1 \in X \\ s_2 \in Y \\ X \cap Y = \emptyset \end{cases} \iff \begin{cases} (1 + \{s_1, s_2\})(XY + X + Y) = 0 \\ \{s_1\}X + \{s_1\} = 0 \\ \{s_2\}Y + \{s_2\} = 0 \\ XY = 0 \end{cases}$$

Let $F = \{(1 + \{s_1, s_2\})(XY + X + Y), \{s_1\}X + \{s_1\}, \{s_2\}Y + \{s_2\}, XY\}$. $\mathbf{B}'$ is a finite subring of $\mathbf{B}$ that consists of $\{0, 1, \{s_1\}, \{s_2\}, \{s_1, s_2\}, 1 + \{s_1\}, 1 + \{s_2\}, 1 + \{s_1, s_2\}\}$. It is isomorphic to $\mathbb{GF}_3$ with the isomorphism ψ given by $\psi(\{s_1\}) = (1, 0, 0)$, $\psi(\{s_2\}) = (0, 1, 0)$

and $\psi(1+\{s_1,s_2\})=(0,0,1)$. *Considering* $\mathbf{B}'$ *as* $\mathbb{GF}_3$ *with this isomorphism,* $F_1=\{0,X+1,0,XY\}$, $F_2=\{0,0,Y+1,XY\}$ *and* $F_3=\{XY+X+Y,0,0,XY\}$. *Reduced Gröbner bases of them in a Boolean polynomial ring* $\mathbb{GF}_2(X,Y)$ *w.r.t. a lexicographic term order such that* $X>Y$ *are* $G_1=\{(1,0,0)Y,(1,0,0)(X+1)\}$, $G_2=\{(0,1,0)(Y+1),(0,1,0)X\}$ *and* $G_3=\{(0,0,1)X,(0,0,1)Y\}$ *respectively. and we have a reduced Boolean Gröbner basis* $G=\{\{s_1\}Y,\{s_1\}(X+1),\{s_2\}(Y+1),\{s_2\}X,(1+\{s_1,s_2\})X,(1+\{s_1,s_2\})Y\}$ *of* F. *Stratified Boolean Gröbner basis is obtained simply adding elements which have the same leading monomial. For this* G, *its stratified Boolean Gröbner basis is* $\{X+\{s_1\},Y+\{s_2\}\}$.

The computation of comprehensive Boolean Gröbner bases is much simpler than the computation of usual comprehensive Gröbner bases in polynomial rings over fields. Given a finite set F of Boolean polynomials in $\mathbf{B}(A_1,\ldots,A_m,X_1,\ldots,X_n)$, let G be a (stratified) Boolean Gröbner basis of the ideal $\langle F\rangle$ in the Boolean polynomial ring $(\mathbf{B}(A_1,\ldots,A_m))(X_1,\ldots,X_n)$ over the coefficient Boolean ring $\mathbf{B}(A_1,\ldots,A_m)$, then G is a (stratified) comprehensive Boolean Gröbner basis of F with parameters $A_1,\ldots,A_m$. We can also apply the above method for them, however, when m is not very small we need a huge natural number k for the isomorphism between $\mathbf{B}(A_1,\ldots,A_m)$ and $\mathbb{GF}_2^k$, namely $k\geq 2^m$. Therefor the above method is not feasible when we have many parameters. The next result recently reported in [2] enables us to apply the above method for the computation of comprehensive Boolean Gröbner bases.

Theorem 3 *Let* $G=\{g_1(\bar{A},\bar{X}),\ldots,g_k(\bar{A},\bar{X})\}$ *be a Boolean Gröbner basis of* $\langle F\rangle$ *for* $F=\{f_1(\bar{A},\bar{X}),\ldots,f_l(\bar{A},\bar{X})\}$ *in a Boolean polynomial ring* $\mathbf{B}(\bar{A},\bar{X})$ *w.r.t. a block term order* $>$ *such that* $\bar{X}\gg\bar{A}$. *Then* G *is a comprehensive Boolean Gröbner basis of* F *w.r.t.* $>_{\bar{X}}$ *(restriction of* $>$ *on* $T(\bar{X})$*).*

Example 2 *The following left constraint is same as the previous example except that we have an another unknown variable a for an element. Using another set variable A to represent a singleton set* $\{a\}$, *it is equivalent to the right system of equations of a Boolean polynomial ring* $\mathbf{B}(A,X,Y)$.

$$\begin{cases} X\cup Y\subseteq\{s_1,s_2\} \\ s_1\in X \\ a\in Y \\ X\cap Y=\emptyset \end{cases} \iff \begin{cases} (1+\{s_1,s_2\})(XY+X+Y)=0 \\ \{s_1\}X+\{s_1\}=0 \\ AY+A=0 \\ XY=0 \end{cases}$$

Let $F=\{(1+\{s_1,s_2\})(XY+X+Y),\{s_1\}X+\{s_1\},AY+A,XY\}$.
The stratified Boolean Gröbner basis G *of* F *w.r.t. a lexicographic term order such that* $X>Y>A$ *has the following form:*

$$G=\{\{s_2\}XY,\{s_2\}YA+\{s_2\}A,(1+\{s_2\})Y,\\ \{s_2\}XA,(1+\{s_2\})X+\{s_1\},(1+\{s_2\})A\}.$$

This is not stratified or even reduced as a Boolean Gröbner basis in $(\mathbf{B}(A))(X,Y)$.
In fact, if we specialize A *by* $\{s_2\}$, $G(\{s_2\})\setminus\{0\}$ *becomes* $\{\{s_2\}XY,\{s_2\}Y+\{s_2\},(1+\{s_2\})Y,\{s_2\}X,(1+\{s_2\})X+\{s_1\}\}$, *which is not stratified or even reduced.*
But, from the above comprehensive Boolean Gröbner basis we can easily construct the stratified Boolean Gröbner basis of F *in* $(\mathbf{B}(A))(X,Y)$*:*

$$\{(\{s_2\}A+\{s_2\})XY,(A+1+\{s_2\})X+\{s_1\}A+\{s_1\},\\ (A+1+\{s_2\})Y+\{s_2\}A,(1+\{s_2\})A\}.$$

3 Implementation

We show how we can implement the computation method of Boolean Gröbner bases described in the last section using the only manipulations of polynomial rings over the Galois field $\mathbb{GF}_2$.

In our implementation, an element of $\mathcal{P}_{FC}(S)$ is represented as a polynomial over $\mathbb{GF}_2$. For example, an element $1 + \{s_1, s_2\}$ of $\mathcal{P}_{FC}(S)$ is represented as a polynomial $1 + s_1 + s_2$ of $\mathbb{GF}_2[s_1, s_2]$. Using this representation, a polynomial f of $\mathcal{P}_{FC}(S)[\bar{X}]$ is translated into a polynomial in $\mathbb{GF}_2[s_1, \ldots, s_t, \bar{X}]$, where $s_1, \ldots, s_t$ are all the strings which occurs in some coefficient of f. The smallest Boolean subring of $\mathcal{P}_{FC}(S)$ that contains $s_1, \ldots, s_t$ is isomorphic to the direct product $\mathbb{GF}_2^{t+1}$. Using an isomorphism ψ such that $\psi(\{s_i\})$ is the $(t+1)$-tuple of $0, 1$ such that only the i-th component is 1 and the others are all 0 and $\psi(1 + \{s_1, \ldots, s_t\})$ is the $(t+1)$-tuple of $0, 1$ such that only the last component is 1, we can consider f as a polynomial of $\mathbb{GF}_2^{t+1}[\bar{X}]$. Under this isomorphism, f_i can be computed by simply specializing s_i with 1 and other s_j with 0 for $i = 1, \ldots, t$, for $i = t+1$ by specializing all variables $s_1, \ldots, s_t$ with 0.

Example 3 *$f = (1 + \{s_1, s_2\})XY$ is translated into $XY + s_1 s_2 XY$.*
$g = \{s_1, s_2\}X(\{s_1\}Y)$ is translated into $(s_1 + s_2)Xs_1Y = s_1^2XY + s_1 s_2 XY$.

Note that the second polynomial g could be further simplified to s_1XY, however we do not employ this simplification since it does not affect the above specializations. By this rather lazy strategy together with the computation technique of Boolean Gröbner bases described in Theorem 1, we can construct most part of Boolean Gröbner bases computations by only using facilities of Risa/Asir. Our codes for the computations of both Boolean Gröbner bases and comprehensive Boolean Gröbner bases consists of less than 300 lines.

Example 4 The following are computation examples of Example 1 and 2 by our software. The first list `[x,y]` is the list of main variables, the second one `[]` or `[a]` is the list of parameters, the third one `[s1,s2]` is the list of strings and the last number `2` is the type of the term order (in Risa/Asir `2` is for a lex order).

```
[1378] G=cbgb([(1+(s1+s2))*(x*y+x+y),s1*x+s1,s2*y+s2,x*y],
                                    [x,y],[],[s1,s2],2)$
[1379] bp_str(G,[x,y],[]);
 [1*y+[s2],1*x+[s1]]

[1380] G=cbgb([(1+(s1+s2))*(x*y+x+y),s1*x+s1,a*y+a,x*y],
                                    [x,y],[a],[s1,s2],2)$
[1382] bp_str(G,[x,y],[a]);
 [([s2]+1)*a,1*a*y+([s2]+1)*y+[s2]*a,1*a*x+([s2]+1)*x+[s1]*a+[s1],
   [s2]*a*y*x+[s2]*y*x]
```

4 Conclusions and remarks

Our program achieves tremendous speed-up comparing with the old implementation of [7]. It enables us to do our recent work [9] of a non-trivial application of boolean Gröbner bases. The following table contains computation time(in terms of seconds) of 7 boolean

Gröbner bases for solving 7 Sudoku puzzles which are ranked as extremely difficult. The row Risa/Asir and Klic contain computation times of the same boolean Gröbner basis in each column by our new program and by the old program of [7] respectively, the symbol ∞ means that the computation did not terminate within 2 hours. All computations are done by a PC with 2GB memory and Core2Duo2GHZ CPU.

puzzle	1	2	3	4	5	6	7
Risa/Asir	41.7	43.6	48.1	40.1	44.3	48.9	76.2
Klic	134.1	398.3	1025.3	∞	1242.3	686.5	∞

References and Notes

[1] Inoue, S.(2009). BGSet - a software to compute Boolean Gröbner bases -. http://www.mi.kagu.tus.ac.jp/ inoue/BGSet

[2] Inoue, S.(2009). On the Computation of Comprehensive Boolean Gröbner Bases. Proceedings of the 11th International Workshop on Computer Algebra in Scientific Computing(CASC 2009), LNCS 5743, pp 130-141, Springer-Verlag Berlin Heidelberg.

[3] Noro, M. et al. (2009). A Computer Algebra System Risa/Asir. http://www.math.kobe-u.ac.jp/Asir/asir.html.

[4] Sakai, K. and Sato, Y. (1988). Boolean Gröbner bases. ICOT Technical Momorandum 488. http://www.icot.or.jp/ARCHIVE/Museum/TRTM/tm-list-E.html

[5] Sakai, K., Sato, Y. and Menju, S. (1991). Boolean Gröbner bases(revised). ICOT Technical Report 613. http://www.icot.or.jp/ARCHIVE/Museum/TRTM/tr-list-E.html

[6] Sato, Y. et al.(1996). Set Constrains Solvers(Prolog version). http://www.icot.or.jp/ARCHIVE/Museum/FUNDING/funding-96-E.html

[7] Sato, Y. et al.(1998). Set Constrains Solvers(Klic version). http://www.icot.or.jp/ARCHIVE/Museum/FUNDING/funding-98-E.html

[8] Sato, Y. and Inoue, S.(2005). On the Construction of Comprehensive Boolean Gröbner Bases. Proceedings of the 7th Asian Symposium on Computer Mathematics(ASCM 2005), pp 145-148.

[9] Sato, Y., Inoue, S., Suzuki, A. and Nabeshima, K. Boolean Gröbner Bases and Sudoku. Submitted for publication.

[10] Sato, Y., Nagai, A. and Inoue, I.(2008). On the Computation of Elimination Ideals of Boolean Polynomial Rings, Proceedings of the 8th Asian Symposium on Computer Mathematics(ASCM 2007), LNAI 5081, pp 338-348, Springer-Verlag Berlin Heidelberg.

[11] Weispfenning, V. (1989). Gröbner bases in polynomial ideals over commutative regular rings. In Davenport Ed., editor, *EUROCAL'87*, pp 336-347. Springer LNCS 378, 1989.

A Symbolic-Numeric Approach to Some Classes of Parametric Optimization Problems for Manufacturing Design

Hidenao Iwane[1*], Hitoshi Yanami[1], Hirokazu Anai[1,2]

[1] FUJITSU LABORATORIES LTD,
4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki 211-8588, Japan
iwane@jp.fujitsu.com, yanami@labs.fujitsu.com

[2] Kyushu University,
744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan
anai@math.kyushu-u.ac.jp

Abstract

We study some important classes of optimization problems originating from optimal design of semiconductor memories such as static random access memory (SRAM) aiming at boosting the yield rate. For the purpose we propose new optimization methods for the classes based on a symbolic algorithm called quantifier elimination (QE) combined with numerical computation. This improves the total efficiency of the design process by reducing the number of repetitions of numerical yield rate simulations and also provides some useful information e.g., explicit relations among design parameters, objective functions and the yield rate.

1 Introduction

Recently, in manufacturing design, model-based design has attracted much attention. Manufacturing design problems can often be treated as mathematical constraints via mathematical models of the target systems. Therefore, developments in design processes are closely related to the available computational methods (in particular, optimization methods) at the time. Numerical convex optimization methods enable us to obtain globally optimal solutions to many design problems that cannot currently be solved analytically. Design methods based on numerical optimizations are becoming more practical due to enhanced computer performance and the development of algorithms with superior accuracy and efficiency. However, some hurdles still remain in these numerically computed design methods. Demands for higher quality, better performance, high-value added, and manufacturing small quantities of a wide variety of products require more accurate globally optimal solutions of non-convex problems as well as parametric solutions to the problems (i.e., regions of feasible solutions, the optimal solutions in terms of the parameters). Constraint solving/optimization methods based on symbolic and algebraic computation have been gaining attention recently as an effective method. Specifically, quantifier elimination (QE), which is an algebraic algorithm based on theories of real algebraic geometry, has been successfully applied in many fields of science and engineering [1, 3, 5]. However, to realize practically effective methods using QE, its speeding-up is the most significant issue.

In this paper we consider some important classes of optimization problems that are originating from an optimal design process for boosting the yield rate of SRAM production. For such optimization problems, various kinds of numerical approaches based on Monte Carlo methods or genetic algorithms (GA) have been commonly used.

*Correspondence to: iwane@jp.fujitsu.com

However, it is desired to develop more efficient methods which help us take an overall view of design due to miniaturization of SRAM technologies. For the purpose, in this paper, we propose new optimization methods based on quantifier elimination by combining it with numerical computation.

2 Statement of the problem

First we explain the target problems which come from SRAM optimal design.

Let $\mathbf{x}$ be a set of variables $\mathbf{x} = (x_1, \ldots, x_n)$ where $\mathbf{x} \in \mathcal{D} \subseteq \mathbb{R}^n$. $\mathcal{D}$ is usually given as a cartesian product of intervals, that is $x_i \in I_i$ where $I_i = [a_i, b_i]$ and $a_i, b_i \in \mathbb{R}$. We consider the three polynomials g_1, g_2 and g_3 in $\mathbf{x}$ over the reals $\mathbb{R}$ and assign new variables y_1, y_2 and y_3 to them respectively: $y_1 \equiv g_1(\mathbf{x})$, $y_2 \equiv g_2(\mathbf{x})$ and $y_3 \equiv g_3(\mathbf{x})$. Then we denote $\min(y_1, y_2)$ by z: $z \equiv \min(y_1, y_2)$. Usually y_1 and y_2 correspond to noise margins of SRAM and z stands for the yield rate of SRAM production. Moreover, y_3 is another objective function such as a cell size of SRAM. To put it simply, what we want to do is to optimize design parameters (e.g., x_i) and objective functions (e.g., y_3) so as to boost the yield rate z. This is regarded as some classes of optimization problems such as parametric optimization or multi-objective optimization. Concretely speaking, fundamental problems to be examined in the design process, which are our main concerns here, are formulated below. Though here we do not mention any combined problems among the following problems, such problems are frequently considered in actual design work.

Problem 1 Find out the maximal value of z:

$$\begin{cases} \text{maximize} & z \equiv \min(y_1, y_2), \\ \text{subject to} & y_1 \equiv g_1(\mathbf{x}),\ y_2 \equiv g_2(\mathbf{x}),\ \ \mathbf{x} \in \mathcal{D} \subseteq \mathbb{R}^n. \end{cases}$$

Problem 2 Find out the relation between z and a design parameter x_i.

Problem 3 Find out the relation between z and another objective function y_3.

In general you may imagine that the polynomials g_1, g_2 and g_3 are nonlinear. However, we should note that we can expect some special structures in the optimization problems derived from the actual situation of SRAM optimal design. In fact, we can assume the following properties for most problems from SRAM optimal design. Our aim is to develop effective and efficient algorithms to solve Problems 1, 2 and 3 by exploiting the following properties.

Structure 1 g_1 and g_2 are polynomial models generated from simulation data. In many cases it is sufficient that we employ linear models w.r.t. $\mathbf{x}$ for g_1 and g_2.

Structure 2 The objective function g_3 associated with a cell size is usually given as a quadratic polynomial in $\mathbf{x}$, which consists of two linear factors $g_3^{(1)}$ and $g_3^{(2)}$: $g_3 = g_3^{(1)} \cdot g_3^{(2)}$.

3 Our solution

The outline of our approach to Problems 1-3 is very simple. Our main tool is a symbolic algorithm "quantifier elimination" for solving real algebraic constraints [2].

⟨1⟩ First each problem is reduced to a QE problem expressed as a first-order formula,
⟨2⟩ then the resulting QE problems are solved by using QE,

⟨3⟩ finally we obtain the feasible regions of desired items (which are chosen at step ⟨1⟩ among design parameters x_i, objective functions y_j, z) in a parameter space.

The solutions to three problems are obtained by solving the following QE problems, respectively. The two structural properties of the problems are incorporated at each step of solving Problems 1, 2 and 3 as shown below:

Problem 1 The associated QE problem is given as

$$\exists \mathbf{x}\ (z \leq g_1(\mathbf{x}) \wedge z \leq g_2(\mathbf{x}) \wedge \mathbf{x} \in \mathcal{D}). \tag{1}$$

Performing QE on (1), we obtain an equivalent quantifier-free formula $\phi(z)$, which describes the feasible regions of z (results in a union of intervals in z). Therefore the desired maximum is the maximal endpoint of all the intervals. Furthermore we can get the exact expression of the maximal value. Usually the polynomials appearing in (1) are all linear, hence we can utilize a specialized QE algorithm [4], which is generally much more efficient than a general QE algorithm, and thus this greatly improves the computational efficiency.

Problem 2 The associated QE problem is given as

$$\exists x_1 \exists x_2 \cdots \exists x_{i-1} \exists x_{i+1} \exists x_{i+2} \cdots \exists x_n\ (z \leq g_1(\mathbf{x}) \wedge z \leq g_2(\mathbf{x}) \wedge \mathbf{x} \in \mathcal{D}). \tag{2}$$

Performing QE on (2), we obtain an equivalent quantifier-free formula $\psi(z, x_i)$, which shows the feasible regions in the z-x_i plane as a semi-algebraic set. By focusing attention on the upper bounds of the feasible regions w.r.t. z, if they exist, one can see the explicit relation between the maximal yield rate and the design parameter x_i. This is a *parametric optimization problem*. If the polynomials appearing in (2) are also all linear, then we can utilize a specialized QE algorithm [4] as well.

Problem 3 The associated QE problem is given as

$$\exists \mathbf{x}\ (z \leq g_1(\mathbf{x}) \wedge z \leq g_2(\mathbf{x}) \wedge y_3 - g_3(\mathbf{x}) = 0 \wedge \mathbf{x} \in \mathcal{D}). \tag{3}$$

Performing QE on (3), we obtain an equivalent quantifier-free formula $\tau(z, y_3)$, which demonstrates the feasible regions in the z-y_3 plane as a semi-algebraic set. By focusing attention on the upper bounds of the feasible regions w.r.t. z and the lower bounds w.r.t. y_3 simultaneously, we can see the relation (typically "trade-off") between maximal yield rate and the cell size y_3. This is a *multi-objective optimization problem*.

As we mentioned, if g_3 is a polynomial with degree 2 in $\mathbf{x}$, it might significantly increase the computational complexity, in particular for the case with many variables. Fortunately, g_3 is normally factorized into two linear factors $g_3^{(1)}$ and $g_3^{(2)}$, so we can reformulate (3) as follows:

$$\exists \mathbf{x}\ (z \leq g_1(\mathbf{x}) \wedge z \leq g_2(\mathbf{x}) \wedge y_4 - g_3^{(1)} = 0 \wedge y_5 - g_3^{(2)} = 0 \wedge \mathbf{x} \in \mathcal{D}). \tag{4}$$

The polynomials occurring in (4) are also all linear, so we can employ a specialized QE algorithm [4] as well. Here we do not have the formula of y_3 explicitly in (4), instead we have the formulas with y_4, y_5. Then we numerically evaluate z, y_4 and y_5 at a sufficiently large number of sample points. We obtain a numerical plot in the z-y_3 plane via $y_3 = y_4 \cdot y_5$. This combined method of QE and numerical sampling is quite efficient and effective compared with conventional methods such as a GA-based method because we can drastically reduce the number of repetitions of numerical yield rate simulations.

Remark 1 *In the actual design field it is required to solve some of the above problems simultaneously, for example, minimize x_1 and y_3 and maximize z concurrently. Such optimizations can be naturally and similarly formulated as QE problems.*

Computational examples: We briefly show computational results of examples for Problems 2 and 3. In Fig.1., (a) is a typical result of the feasible region in z-x_1, here x_1 corresponds to a voltage. The result for Problem 3 obtained by our method after 10,000 times of simulations is (b) and the result for the same problem obtained by a GA-based method after 20,000 times of simulations is (c). The figure (d) is the superposition of (b) and (c). We used a GA tool called *Single and Multiobjective Genetic Algorithm Toolbox**, which is developped by K. Sastry, for computing (c). We can say that (c) is less optimal compared with (b) (see the portion indicated by a circle in (c)).

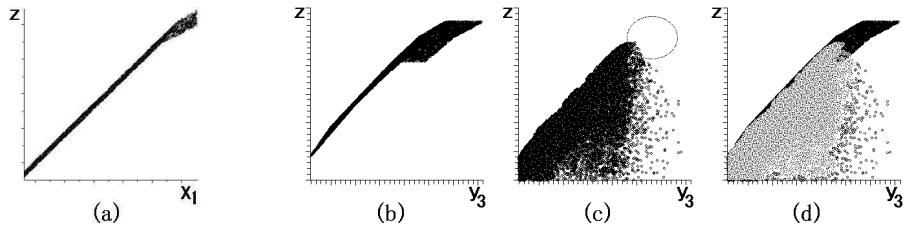


Figure 1: Computational results for Problems 2 and 3

4 Conclusions

We have proposed new optimization methods based on quantifier elimination combined with numerical computation for some important classes of optimization problems in SRAM optimal design. The methods improve the total efficiency of the design process by reducing the number of repetitions of numerical simulations and also bring some useful information e.g., relations among design parameters or objective functions.

References and Notes

[1] H. Anai, S. Hara, M. Kanno, and K. Yokoyama. Parametric polynomial spectral factorization using the sum of roots and its application to a control design problem. *J. Symb. Comput.*, 44(7): 703-725, 2009.

[2] B. Caviness and J. Johnson, editors. Quantifier elimination and cylindrical algebraic decomposition. Texts and monographs in symbolic computation. Springer-Verlag, 1998.

[3] T. Sturm. New domains for applied quantifier elimination. In Proceedings of CASC 2006. pp295–301, 2006.

[4] V. Weispfenning. The complexity of linear problems in fields. *J. Symb. Comput.*, 5(1–2):3–27, Feb.–Apr. 1988.

[5] H. Yanami and H. Anai. The Maple package SyNRAC and its application to robust control design. *Future Generation Comp. Syst.*, 23(5):721–726, 2007.

*See `http://www.kumarasastry.com/2007/06/11/single-and-multiobjective-genetic-algorithm-toolbox-in-c/`

Design of a PI controller with H_∞ performance and step response constraints

Takuya Kitamoto[1*] and Tetsu Yamaguchi[2]

[1] Faculty of Education, Yamaguchi University, 1-1677 Yoshida, Yamaguchi Japan
kitamoto@yamaguchi-u.ac.jp

[2] Cybernet Systems, Co, LTD, Tokyo, Japan
tetsuy@cybernet.co.jp

Abstract

Recently, Computer algebra system (CAS) such as Maple, Mathematica is gaining its popularity in various fields of science, education and engineering. Symbolic computation, one of their features, provides us new applications of computer systems that conventional numerical packages can not. One of examples of such applications is control engineering, where unknown parameters play an important role as design parameters and uncertain indeterminates. This paper utilizes the CAS to PID control theory (PID control is a basis of feedback control theory, and despite of celebrated modern control theory, has been a major force in industrial applications). We focus on a PI controller that has the form $k_p + \frac{k_i}{s}$ with two real parameters $k_p,\ k_i \in \mathbf{R}$. Although the form of the controller is rather simple, design of parameters $k_p,\ k_i \in \mathbf{R}$ is usually difficult, and often requires trial and error process (in general, there is no clear link between parameter values and desired properties of a controlled system). This paper presents a method to design the parameters $k_p,\ k_i$ so that H_∞ norm $|G(s)|_\infty$ is minimized where $G(s)$ is the transfer function from reference r to output y of the controlled system. Such control is called 'H_∞ optimal control' in modern control theory, and our method can be viewed as a mixture of classical and modern control theory. We also consider constraints on the step response and frequency restricted norm of the system, and present a method to design the parameters $k_p,\ k_i$ satisfying those constraints.

1 Introduction

Recently, Computer algebra system (CAS) such as Maple, Mathematica is gaining its popularity in various fields of science, education and engineering. Symbolic computation, one of their features, provides us new applications of computer systems that conventional numerical packages can not. One of examples of such applications is control engineering, where unknown parameters play an important role as design parameters and uncertain indeterminates. For example, [1]-[6] apply Quantifier elimination (QE), which is a new technique of computer algebra, to the design of control systems. In [7]-[9], using a similar technique to QE, the authors of this paper present methods to compute H_∞ norm of a given parametric system (system with a parameter) as a root of polynomial. The algorithms in [7]-[9] are extended in [12] so that it can compute frequency restricted norm (FRN), which is a generalization of H_∞ norm to a finite range of frequencies. In [11], the same authors present a method to compute H_2 norm of a given parametric system.

In this paper, we focus on a PI controller that has the form $k_p + \frac{k_i}{s}$ with two real parameters $k_p,\ k_i \in \mathbf{R}$. PID control theory has a long history and is formulated in the framework of classical control. It is a basis of feedback control theory, and despite of celebrated modern

*Correspondence to: Yamaguchi University, 1-1677 Yoshida, Yamaguchi, Japan Tel: +81-83-933-5336

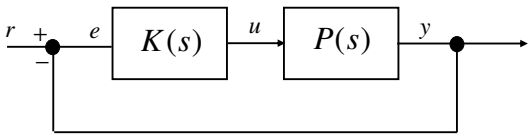


Figure 1: Control system

control theory, has been a major force in industrial applications. Although the form of the PI controller is simple, design of such controllers usually requires trial and error process. Because it is often hard to determine parameter values $k_p,\ k_i \in \mathbf{R}$ so that controlled system has desired properties (usually there is no clear link between desired properties of a controlled system and parameter values).

This paper presents a method to design parameters k_p, k_i so that H_∞ norm $|G(s)|_\infty$ of $G(s)$ is minimized where $G(s)$ is the transfer function from reference r to output y of the controlled system. We also consider constraints on the step response and frequency restricted norm of the system, and present a method to design the parameters $k_p,\ k_i$ satisfying those constraints. We demonstrate the methods in this paper, showing illustrative design examples of a control system.

This paper is composed as follows: In Section 2, we formulate our problem. Then, in Section 3, we describe how to optimize the parameters according to given specifications. In Section 4, we show two design examples, and lastly, in Section 5, we conclude.

2 Problem formulation

Let us consider the system in Fig. 1, where $P(s)$ is a given plant with one input and one output and $K(s)$ is a PI controller in the form of $K(s) = k_p + \frac{k_i}{s}$. In this paper, we consider the following conditions to determine design parameters $k_p,\ k_i \in \mathbf{R}$:

(C1) The controlled system is stable, i.e. all roots of the denominator of $1/(P(s)K(s)+1)$ have negative real parts.

(C2) H_∞ norm $|G(s)|_\infty$ is minimized where $G(s)$ is the transfer function from r to y in Fig. 1 and H_∞ norm of $G(s)$ is defined by

$$|G(s)|_\infty \stackrel{\text{def}}{=} \sup_{0\le\omega} |G(i\omega)|. \tag{1}$$

(C3) Frequency restricted norm $\mathrm{FRN}(G(s))_{[\underline{\omega},\overline{\omega}]}$ is minimized where $G(s)$ is the transfer function from r to y in Fig. 1 and frequency restricted norm of $G(s)$ is defined by

$$\mathrm{FRN}(G(s))_{[\underline{\omega},\overline{\omega}]} \stackrel{\text{def}}{=} \sup_{\underline{\omega}\le\omega\le\overline{\omega}} |G(i\omega)|. \tag{2}$$

(C4) Rise time (the time required for $y(t)$ to change from $y(t) = 0$ to $y(t) = 0.9$ to step input) of the controlled system is minimized.

3 Optimization of design parameters

3.1 Equivalent condition for condition (C1)

We can apply well-known Routh stability criteria to the denominator of $1/(P(s)K(s)+1)$, which provides us an equivalent condition for condition (C1) in the form of

$$f_1(k_p,k_i)>0,\ f_2(k_p,k_i)>0,\ \cdots,\ f_n(k_p,k_i)>0, \tag{3}$$

where $f_j(k_p,k_i)$ are rational functions in k_p and k_i.

3.2 Equivalent condition for condition (C2)

First, we express H_∞ norm $|G(s)|_\infty$ as a root of a polynomial with algorithms in [8], which computes polynomial $f(k_p,k_i,q)$ satisfying for any k_p,k_i that

$$f(k_p,k_i,q)=0,\ q=\left(1/|G(s)|_\infty\right)^2. \tag{4}$$

Obviously, the minimization problem of $|G(s)|_\infty$ is equivalent to the maximization problem of q in (4). We note that the gradient of q with respect to k_i and k_p can be computed as follows: Suppose that $f(k_p,k_i,q)$ is in the form of

$$f(k_p,k_i,q)=f_0(k_p,k_i)+f_1(k_p,k_i)q+\cdots+f_r(k_p,k_i)q^r, \tag{5}$$

where $f_j(k_p,k_i)$ $(j=1,\cdots,r)$ are polynomials in k_p,k_i. Differentiating the above equation by k_p, we obtain

$$\frac{\partial f_0(k_p,k_i)}{\partial k_p}+\cdots+\frac{\partial f_r(k_p,k_i)}{\partial k_p}q^r+\left\{f_1(k_p,k_i)+\cdots+rf_r(k_p,k_i)q^{r-1}\right\}\frac{\partial q}{\partial k_p}=0. \tag{6}$$

Thus, we see that

$$\frac{\partial q}{\partial k_p}=-\frac{\frac{\partial f_0(k_p,k_i)}{\partial k_p}+\cdots+\frac{\partial f_r(k_p,k_i)}{\partial k_p}q^r}{f_1(k_p,k_i)+\cdots+rf_r(k_p,k_i)q^{r-1}}. \tag{7}$$

Similarly, we obtain

$$\frac{\partial q}{\partial k_i}=-\frac{\frac{\partial f_0(k_p,k_i)}{\partial k_i}+\cdots+\frac{\partial f_r(k_p,k_i)}{\partial k_i}q^r}{f_1(k_p,k_i)+\cdots+rf_r(k_p,k_i)q^{r-1}}. \tag{8}$$

Thus, the gradient of q with respect to k_p,k_i is given by

$$\begin{bmatrix}\frac{\partial q}{\partial k_p}\\ \frac{\partial q}{\partial k_i}\end{bmatrix}=\frac{1}{F(k_p,k_i,q)}\begin{bmatrix}\frac{\partial f_0(k_p,k_i)}{\partial k_p}+\cdots+\frac{\partial f_r(k_p,k_i)}{\partial k_p}q^r\\ \frac{\partial f_0(k_p,k_i)}{\partial k_i}+\cdots+\frac{\partial f_r(k_p,k_i)}{\partial k_i}q^r\end{bmatrix},$$
$$F(k_p,k_i,q)=-\left\{f_1(k_p,k_i)+\cdots+rf_r(k_p,k_i)q^{r-1}\right\}. \tag{9}$$

We can apply the above gradient to a numerical optimization package to compute the maximum of q. We can also use the following method to compute all the extrema of q: Since (9) implies that extrema of q with respect to k_p,k_i satisfy

$$\begin{cases}\frac{\partial f_0(k_p,k_i)}{\partial k_p}+\cdots+\frac{\partial f_r(k_p,k_i)}{\partial k_p}q^r=0,\\ \frac{\partial f_0(k_p,k_i)}{\partial k_i}+\cdots+\frac{\partial f_r(k_p,k_i)}{\partial k_i}q^r=0,\\ f(q,k_p,k_i)=0,\end{cases} \tag{10}$$

any extremum of q is a root of the above system of polynomial equations. Hence, computing the roots of (10), 5 with Groebner basis, we obtain all the extrema of q with respect to k_p, k_i.

3.3 Equivalent condition for condition (C3)

Reference [12] presents an algorithm to compute FRN (2) as a root of polynomial. More concretely, the algorithm in [12] computes polynomial $f(k_p, k_i, q)$ satisfying for any k_p, k_i that

$$f(k_p, k_i, q) = 0,\ q = \left(1/\mathrm{FRN}(G(s))_{[\underline{\omega},\overline{\omega}]}\right)^2, \tag{11}$$

where $\underline{\omega}, \overline{\omega}$ are given real numbers. The minimization of $\mathrm{FRN}(G(s))_{[\underline{\omega},\overline{\omega}]}$ can be performed as is done in the previous subsection.

3.4 Equivalent condition for condition (C4)

Step response, which is the time response to step input

$$r = \begin{cases} 1 & \text{when } t > 0 \\ 0 & \text{when } t \leq 0 \end{cases},$$

of output $y(t)$ is given by $y(t) = \bar{C}\bar{A}^{-1}(e^{\bar{A}t} - E)\bar{B}$ where E denotes the identity matrix and $(\bar{A}, \bar{B}, \bar{C})$ is a state space realization of $G(s) = \frac{P(s)K(s)}{P(s)K(s)+1}$. This implies that the rise time t_0 (i.e. the time satisfying $y(t_0) = 0.9$) is a root of equation

$$\bar{C}\bar{A}^{-1}(e^{\bar{A}t} - E)\bar{B} = 0.9$$

with respect to t. When matrices $\bar{A}, \bar{B}, \bar{C}$ contain parameters k_p, k_i, roots of the above equation are functions of k_p, k_i, and it is difficult to compute the functions in explicit form in general. However, as is shown in [13], it is possible to compute the functions in the form of power series of k_p, k_i as follows: First, let us define $\bar{y}(t)$ by $\bar{y}(t) = y(t) - 0.9$. Then, t_0 is a root of $\bar{y}(t) = 0$, and its power series can be computed by the following symbolic Newton's method:

$$t_0^{(2^{k+1}-1)} \leftarrow t_0^{(2^k-1)} - \frac{\bar{y}(t_0^{(2^k-1)})}{\frac{d\bar{y}}{dt}(t_0^{(2^k-1)})} \pmod{(k_p, k_i)^{2^{k+1}}} \tag{12}$$

In the above formula, the arithmetic is performed as arithmetic of power series, and $t_0^{(r)}$ denotes power series expansion of $t_0^{(r)}$ up to r-th total degree of k_p and k_i (for details of symbolic Newton's method, see [15],[16]). Since $\bar{y}(t)$ and $\frac{d\bar{y}}{dt}$ are given by

$$\bar{y}(t) = \bar{C}\bar{A}^{-1}(e^{\bar{A}t} - E)\bar{B} - 0.9,\quad \frac{d\bar{y}}{dt}(t) = \bar{C}e^{\bar{A}t}\bar{B}, \tag{13}$$

the only difficulty of performing the above symbolic Newton's method is the evaluation of $e^{\bar{A}t^{(2^k-1)}}$ as a power series up to $(2^{k+1}-1)$-th total degree in k_p and k_i. Reference [13] presents a method to evaluate the function by matrix pade approximation of the exponential function, and we can compute the power series expansion $t_0^{(m)}$ of t_0 up to any total degree m of k_p and k_i. In this way, an approximation of $t_0^{(m)}$ of the rise time t_0 can be computed, and condition (C4) can be approximated by the minimization of polynomial $t_0^{(m)}$.

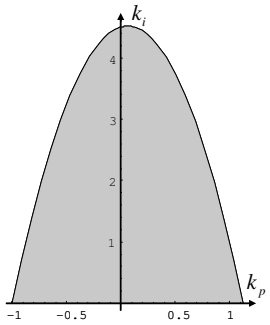


Figure 2: The region of parameters satisfying condition (C1)

4 Design examples

4.1 Design example I

Let us consider the system given by $P(s) = \frac{16}{(s+1)(s^2+s+16)}$. In this subsection, we design a PI controller $K(s) = k_p + \frac{k_i}{s}$ that satisfies conditions (C1), (C2) and (C4).

First, let us derive an equivalent condition for (C1). Since the denominator of $\frac{1}{1+P(s)K(s)}$ is given by $g(s) = s^4 + 2s^3 + 17s^2 + 16(k_p+1)s + 16k_i$, we apply Routh stability criteria to $g(s)$ and obtain

$$9 - 8k_p > 0,\ \frac{16(2k_i + 8k_p^2 - k_p - 9)}{8k_p - 9} > 0,\ 16k_i > 0, \tag{14}$$

which is the equivalent condition for $g(s)$ to be stable (all roots have negative real parts). Obviously, the condition in (14) is equivalent to

$$-2k_i - 8k_p^2 + k_p + 9 > 0,\ k_i > 0, \tag{15}$$

which is the region displayed in Fig. 2. Although condition (C1) is satisfied on the any point in the region in Fig. 2, the point near the boundary of the region is on the verge of instability and undesirable. Hence, we set stricter condition,

$$-2k_i - 8k_p^2 + k_p + 9 > 0.5,\ k_i > 0.5, \tag{16}$$

which is the region displayed in Fig. 3, to guarantee the robustness of the controlled system.

Next, let us consider condition (C2). Transfer function $G(s)$ from r to y is given by

$$G(s) \quad = \quad \frac{P(s)K(s)}{1 + P(s)K(s)} = \frac{16(k_i + k_p s)}{16k_i + (16 + 16k_p)s + 17s^2 + 2s^3 + s^4}, \tag{17}$$

whose state space realization $(\bar{A}, \bar{B}, \bar{C})$ is

$$\bar{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -16k_i & -(16k_p + 16) & -17 & -2 \end{bmatrix},\quad \bar{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$
$$\bar{C} = \begin{bmatrix} 16k_i & 16k_p & 0 & 0 \end{bmatrix}. \tag{18}$$

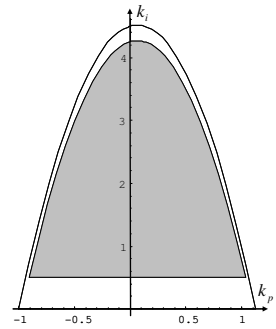


Figure 3: The region of parameters satisfying (16)

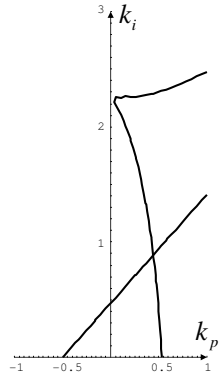


Figure 4: Plot of zeros of (20)

With the algorithm in [8], polynomial $f(q, k_p, k_i)$ satisfying (4) is computed to be

$$f(q, k_p, k_i) = (q-1)(1306368 - 6542208k_i + 3271104k_p + \cdots + 442368k_p^8 q^4). \quad (19)$$

It is easy to see that $|G(s)|_\infty \geq 1$, because we have $G(0) = 1$. In this case, we have the region of the parameters that attains the minimum $|G(s)|_\infty = 1$. To compute the region of parameters, we substitute $q = 1$ into the second factor of (19), and obtain

$$(-8 + 17k_i - 16k_p)^2(20412 - 15471k_i + \cdots - 4096k_p^3). \quad (20)$$

We show the plot of zeros of (20) ((k_p, k_i) satisfying equation (20) $= 0$) in Fig. 4. From the continuity of H_∞ norm, the region of parameters satisfying $|G(s)|_\infty = 1$ must be encircled by the zeros of (20). In fact, the region is given by the shaded region of Fig. 5. In other words, all parameters in the shaded region in Fig. 5 attains the minimum H_∞ norm $|G(s)|_\infty = 1$.

From Fig. 3 and 5, we see that all parameters in the shaded region in Fig. 6 satisfy conditions (C1) and (C2), and let us consider condition last condition (C4). We first compute the rise time t_0 as a power series in k_p and k_i with expansion point $(k_p, k_i) = (0.4, 0.8)$, using technique in [13]. Fig. 7 shows the step response of $y(t)$ when $(k_p, k_i) = (0.4, 0.8)$. From the figure, we see that constant term of t_0 (i.e. $t_0|_{(k_p,k_i)=(0.4,0.8)}$) is 2.50837.

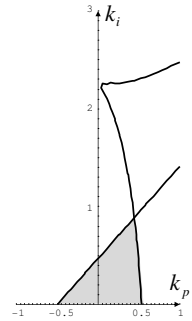


Figure 5: The region of parameters satisfying $|G(s)|_\infty = 1$

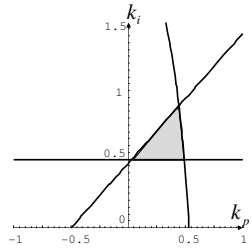


Figure 6: The region of parameters satisfying (16) and $|G(s)|_\infty = 1$

Thus we set $t_0^{(0)} = 2.50837$. Then, we compute $t_0^{(2^k)}$ with (12) where $\bar{y}(t)$ is given by (13) and (18), and obtain

$$\begin{aligned} t_0^{(3)} &= 2.50837 - 1.23291\bar{k}_i - 0.167303\bar{k}_p + 1.71757\bar{k}_i^2 + 2.82468\bar{k}_i\bar{k}_p - 0.521985\bar{k}_p^2 \\ &\quad -6.58699\bar{k}_i^3 - 2.23265\bar{k}_i^2\bar{k}_p - 1.54157\bar{k}_i\bar{k}_p^2 + 0.546237\bar{k}_p^3, \end{aligned} \tag{21}$$

where $\bar{k}_p = k - 0.4$ and $\bar{k}_i = k_i - 0.8$. The above $t_0^{(3)}$ is Taylor series expansion of the rise time t_0, and hence is an approximation of t_0. Thus, the parameter satisfying the conditions (C1), (C2) and (C4) is given by solving the optimization problem $\max_{(k_p,k_i)\in\Omega_1} t_0^{(3)}$ where Ω_1 is the shaded region in Fig. 6. The solution of the above optimization problem is computed to be

$$\max_{(k_p,k_i)\in\Omega_1} t_0^{(3)} = 2.40941 \tag{22}$$

with $(k_p, k_i) = (0.44444, 0.88889)$. Fig. 8 shows the step response of $y(t)$ when $(k_p, k_i) = (0.44444, 0.88889)$. From the figure, we see that the rise time t_0 is improved to $t_0 = 2.41073$. Comparing this with (22), we also see that the relative error in approximation $t_0^{(3)}$ is

$$|2.41073 - 2.40941|/2.41073 = 0.055\%$$

and $t_0^{(3)}$ is a good enough approximation of t_0.

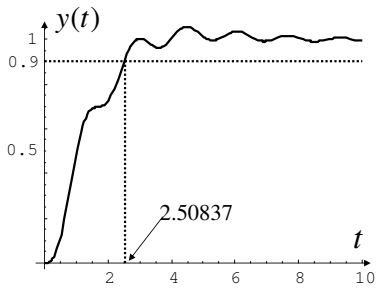


Figure 7: Step response with $(k_p, k_i) = (0.4, 0.8)$

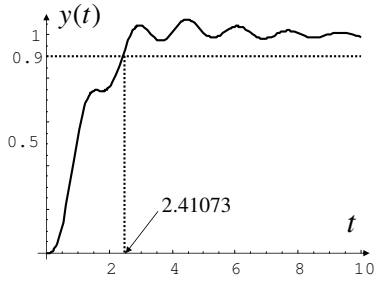


Figure 8: Step response with $(k_p, k_i) = (0.44444, 0.88889)$

4.2 Design example II

Let plant $P(s)$ be the same as in Design example I, and consider conditions (C1) and (C3) with $\underline{\omega} = 1$, $\overline{\omega} = \infty$. Condition (C1) is satisfied by the parameters in the shaded region in Fig. 2, and we use stricter condition (16) that is shown in Fig. 3 as we did in Design example I.

Next, let us consider condition (C3). Unlike H_∞ norm $|G(s)|_\infty$, $\mathrm{FRN}(G(s))_{[1,\infty]}$ can be less than 1. In fact, with $(k_p, k_i) = (0, 0.5)$, we have $\mathrm{FRN}(G(s))_{[1,\infty]} = 0.496139$. Thus, we consider minimization problem

$$\min_{(k_p,k_i)\in\Omega_2} \mathrm{FRN}(G(s))_{[1,\infty]}, \tag{23}$$

where Ω_2 is the shaded region in Fig. 3. First, we compute polynomial $f(q, k_p, k_i)$ satisfying (11) with the algorithm in [12], and obtain

$$f(q, k_p, k_i) = (q-1)(1306368 + \cdots + 442368k_p^8q^4)(-113 + \cdots + 64k_p^2q). \tag{24}$$

First two factors in the right-hand side of (24) is the same as ones in (19), hence the above $f(q, k_p, k_i)$ can be written as

$$\{\text{Polynomial in (19)}\} \times (-113 + 128k_i + \cdots + 64k_p^2q). \tag{25}$$

Since we have (11), minimization problem (23) is equivalent to the maximization problem

$$\max_{(k_p,k_i)\in\Omega_2} \{q \text{ in (11)}\}.$$

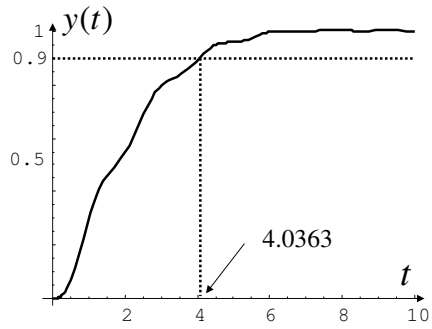


Figure 9: Step response with $(k_p, k_i) = (0.22688, 0.5)$

To compute extrema of q in (11), we compute roots of (10) satisfying $(k_p, k_i) \in \Omega_2$, and find that there is no root corresponding to $\mathrm{FRN}(G(s))_{[1,\infty]}$ inside of Ω_2. Thus, the maximum is taken on the boundary of Ω_2. Searching the maximum on the boundary of Ω_2, we find that

$$\max_{(k_p,k_i)\in\Omega_2} \{q \text{ in } (11)\} = 4.85659, \tag{26}$$

which is taken at $(k_p, k_i) = (0.22688, 0.5)$. Hence, we see

$$\min_{(k_p,k_i)\in\Omega_2} \mathrm{FRN}(G(s))_{[1,\infty]} = 1/\sqrt{4.85659} = 0.453768. \tag{27}$$

For the reference, we show the step response of $y(t)$ with $(k_p, k_i) = (0.22688, 0.5)$ in Fig. 9. The figure shows the rise time is 4.0363 in this case.

5 Conclusion

We proposed a method to design a PI controller $K(s) = k_p + \frac{k_i}{s}$ that considers conditions (C1), (C2), (C3) and (C4). It is shown that condition (C1) is equivalent to a set of polynomial inequalities, and condition (C2) and (C4) are equivalent to the maximization of a polynomial root. Condition (C4) can be approximated by a polynomial minimization. Utilizing these equivalent conditions, a controller design is formulated as a minimization problem with polynomial inequality constraints, which can be solved by computing polynomial roots.

Two design examples are shown. In the first design example, we design a PI controller satisfying condition (C1), (C2) and (C4) for a given system, and obtain a controller such that (i) $|G(s)|_\infty = 1$, and (ii) rise time of the controlled system is 2.41073. In the second design example, we consider condition (C1) and (C3), and design a PI controller that minimizes FRN of $G(s)$ (the minimized FRN is 0.453768).

References and Notes

[1] C. Abdallah, P. Dorato, W. Yang, R. Liska and S. Steinberg, "Application of Quantifier Elimination Theory to Control System Design," *Proc. of 4th IEEE Mediterranean Symposium of Control and Automation* pp. 340–345, Maleme, Crete, 1996.

[2] H. Anai and H. Yanami, "SyNRAC: A maple-package for solving real algebraic constraints," *Proc. of CASA'2003*, P.M.A. Sloot et al. (ICCS 2003) editors, Vol. 2657 of LNCS, Springer-Verlag, 2003.

[3] H. Anai and P. A. Parrilo, "Convex quantifier elimination for semidefinite programming," *In Proc. of CASC'2003*, pp. 3-11, Passau, Germany, 2003.

[4] P. Dorato, W. Yang and C. Abdallah, "Robust Multi-Objective Feedback Design by Quantifier Elimination," *J. Symbolic Computation*, Vol. 24, pp. 153–159, 1997.

[5] H. Hong, R. Liska and S. Steinberg, "Testing Stability by Quantifier Elimination," *J. Symbolic Computation*, Vol. 24, pp. 161–187, 1997.

[6] I. A. Fotiou, P. Rostalski, P. A. Parrilo, and M. Morari, "Parametric optimization and optimal control using algebraic geometry methods," *International Journal of Control*, Vol. 79, No. 11, pp. 1340-1358, 2006.

[7] T. Kitamoto. On the computation of H_∞ norm of a system with a parameter. *The IEICE Trans. Funda. (Japanese Edition)*, Vol. J89-A, No. 1, pp. 25–39, 2006.

[8] T. Kitamoto and T. Yamaguchi. Parametric Computation of H_∞ Norm of a System. *Proc. SICE-ICCAS2006*, Busan, Korea, 2006.

[9] T. Kitamoto and T. Yamaguchi. The optimal H_∞ norm of a parametric system achievable using a static feedback controller. *The IEICE Trans. Funda.*, Vol. E90-A, No. 11, pp. 2496–2509, 2007.

[10] T. Kitamoto and T. Yamaguchi. The optimal H_∞ norm of a parametric system achievable using a output feedback controller. *The IEICE Trans. Funda.*, Vol. E91-A, No. 7, pp. 1713–1724, 2008.

[11] T. Kitamoto and T. Yamaguchi. On the parametric LQ control problem. *The IEICE Trans. Funda. (Japanese Edition)*, Vol. J91-A, No. 3, pp. 349–359, 2008.

[12] T. Kitamoto. Extension of the algorithm to compute H_∞ norm of a parametric system. *The IEICE Trans. Funda.*, Vol. E92-A, No. 8, pp. 2036-2024, 2009.

[13] T. Kitamoto. Computation of the Peak of Time Response in the Form of Formal Power Series. *The IEICE Trans. Funda.*, Vol. E86-A, No. 12, pp. 3240–3250, 2003.

[14] K. Zhou, J. Doyle and K. Glover, "Robust and Optimal Control," *Prentice-Hall. Inc*, New Jersey, 1996.

[15] K.O. Geddes, S.R. Czapor and G. Labahn, "Algorithms for Computer Algebra," Kluwer Academic Publishers, Boston, 1992.

[16] J. Gathen and J. Gerhand, "Modern Computer Algebra," Cambridege University Press, Cambridge, 1999.

Comprehensive Gröbner Bases in a Java Computer Algebra System

Heinz Kredel
IT-Center, University of Mannheim, 68131 Mannheim, Germany
kredel@rz.uni-mannheim.de

Abstract

We present an implementation of the algorithms for computing comprehensive Gröbner bases in a Java computer algebra system (JAS). Contrary to approaches to implement comprehensive Gröbner bases with minimal requirements to the computer algebra system, we aim to provide and utilize all necessary algebraic structures occurring in the algorithm. In the implementation of a condition we aim at the maximal semantic exploitation of the occurring algebraic structures: the set of equations equal zero are implemented as an ideal (with Gröbner base computation) and the set of inequalities are implemented as a multiplicative set which is simplified to polynomials of minimal degrees using, for example, square-free decomposition. With our approach we can also make the transition of a comprehensive Gröbner system to a polynomial ring over a (commutative, finite, von Neuman) regular coefficient ring and test or compute Gröbner bases in such polynomial rings.

1 Introduction

In this paper we present an implementation of the algorithms for computing comprehensive Gröbner bases [23, 25, 14, 20] in a Java computer algebra system (JAS) [5, 9, 6, 7].

JAS uses Java to implement a computer algebra library with special emphasis on object oriented programming in an algebraic setting. The emphasis of this paper is also on the library design for comprehensive Gröbner bases. Contrary to approaches to implement comprehensive Gröbner bases with minimal requirements to the computer algebra system, like the one of Suzuki and Sato [21], we aim to provide and utilize all necessary algebraic structures occurring in the algorithm. For example there are parametric polynomials, colored polynomials or coefficients in residue class rings.

In the implementation of a condition we aim at the maximal semantic exploitation of the occurring algebraic structures: the set of equations equal zero are implemented as an ideal (with Gröbner base computation and ideal membership test) and the set of inequalities are implemented as a multiplicative set which is simplified to polynomials of minimal degrees using square-free decomposition or factorization. This approach has partially been taken by [19, 12, 1].

With our approach we can even make the transition of a comprehensive Gröbner system to a polynomial ring over a (commutative, finite, von Neuman) regular coefficient ring and test or compute Gröbner bases in such polynomial rings [22, 24].

1.1 Related Work

Comprehensive Gröbner bases have been introduced by Weispfenning [23] and improved to obtain canonical properties in [25]. Further improvements are achieved by Montes and Manubens [14, 15] and alternative approaches are presented by Sato and Suzuki [20, 21] and [4].

A first implementation comprehensive Gröbner bases was by [19] in ALDES/SAC-2 and MAS, which was improved in [12] and [1]. Newer implementations are presented in [16, 3, 21].

Due to limited space we do not discuss the related mathematical work on Gröbner bases and other computer algebra algorithms, which can be found in standard text books.

1.2 Outline

In the next section 2, we give an an example on using the JAS library. Due to limited space we must assume that you are familiar with the Java programming language, object oriented programming and the JAS type system [9, 11]. Section 3 presents the design of the classes for the implementation of comprehensive Gröbner bases. The topics of the subsections are: conditions and colored polynomials, parametric reduction and colored systems, Gröbner systems and comprehensive Gröbner bases. In section 4 we present some examples with performance measurements and the transition to regular coefficient rings. Finally section 5 draws some conclusions.

2 Introduction to the JAS Library

In this section we discuss an example for the usage of the JAS library. This section contains revised parts of the JAS introduction in [8].

JAS provides a well designed library for algebraic computations implemented with the aid of Java's generic types. The library can be used as any other Java software package or it can be used interactively or interpreted through an jython (Java Python) front end. JAS implements interfaces and classes for basic arithmetic of arbitrary precision integers, rational numbers and multivariate polynomials with such coefficients. Other packages in JAS are: `edu.jas.ufd` with algorithms for unique factorization domains. `edu.jas.gb` with classes for polynomial and solvable polynomial reduction, Gröbner bases and ideal arithmetic as well as thread parallel and distributed versions of Buchberger's algorithm. `edu.jas.gbmod` contains classes for module Gröbner bases, syzygies for polynomials and solvable polynomials.

For an introduction to the JAS type system see [9, 11]. To get an idea of the interplay of the types, classes and object construction consider the following type

```
List<GenPolynomial<Product<Residue<BigRational>>>>
```

of a list of polynomials with coefficients from a direct product of residue class rings modulo some polynomial ideal over the rational numbers. It arises in the computation of Gröbner bases over commutative regular rings $S' = \left(\prod_{\mathfrak{p}\in \mathrm{spec}(R)} R/\mathfrak{p}\right)[y_1, \ldots, y_r]$, where $R = \mathbb{Q}[x_1, \ldots, x_n]$, see [17, 22, 24] and section 4. To keep the example simple we will show how to generate a list `L` of polynomials in the ring

$$S = (\mathbb{Q}[x_0, x_1, x_2]/\mathrm{ideal}(F))^4[a, b].$$

The ring S is represented by the object in variable `fac` in the listing in figure 2. Random polynomials of this ring may look like the one shown in figure 1. The coefficients from $(\mathbb{Q}[x_0, x_1, x_2]/\text{ideal}(F))^4$ are shown enclosed in braces `{}` in the form `i=polynomial`. I.e. the index `i` denotes the product component $i = 0, 1, 2, 3$ which reveals that the `Product` class is implemented using a sparse data structure. The list of F is printed after the '`rr =`' together with the indication of the type of the residue class ring `ResidueRing` as polynomial ring in the variables `x0, x1, x2` over the rational numbers `BigRational` with graded lexicographical term order `IGRLEX`. The variables `a, b` are from the 'main' polynomial ring and the rest of figure 1 should be obvious.

```
rr = ResidueRing[ BigRational( x0, x1, x2 ) IGRLEX
     ( ( x0^2 + 295/336  ),
     ( x2 - 350/1593 x1 - 1100/2301 ) ) ]
L = [ {0=x1 - 280/93 , 2=x0 * x1 - 33/23 } a^2 * b^3
    + {0=122500/2537649 x1^3 + 770000/3665493 x1^2
      + 14460385/47651409 x1 + 14630/89739 ,
      3=350/1593 x1 + 23/6 x0 + 1100/2301 } , ... ]
```

Figure 1: Random polynomials from ring S

The output in figure 1 is computed by the program from figure 2. Line number 1 defines the variable `L` of our intended type and creates it as an Java `ArrayList`. Lines 2 and 3 show the creation of the base polynomial ring $\mathbb{Q}[x_0, x_1, x_2]$ in variable `pfac`. In lines 4 to 9 the list `F` of random polynomials is constructed which will generate the ideal of the residue class ring. Lines 10 to 13 create a Gröbner basis for the ideal, setup the residue class ring `rr` and print it out. Line 14 constructs the regular ring `pr` as direct product of 4 copies of the residue class ring `rr`. The the final polynomial ring `fac` in the variables `a` and `b` is defined in lines 15 and 16. Lines 17 to 22 then generate the desired random polynomials, put them to the list `L` and print it out.

With this example we see that the software representations of rings snap together like 'LEGO blocks' to build up arbitrary structured rings. This concludes the introduction to JAS, further details can be found, as already mentioned, in [11, 5, 7, 9, 8].

3 Comprehensive Gröbner Bases

Recall some definitions from [23]. Let K be a field, $R = K[U_1, \ldots, U_m]$ a polynomial ring over K in the variables $U_1, \ldots, U_m$. Let $S = R[X_1, \ldots, X_n]$ be a polynomial ring over R in the variables $X_1, \ldots, X_n$ and let $\prec_S$ be a term order on S. S is called a parametric polynomial ring with parameters $U_1, \ldots, U_m$ in the main variables $X_1, \ldots, X_n$. $K[U_1, \ldots, U_m][X_1, \ldots, X_n]$ will be abbreviated by $K[\mathbf{U}][\mathbf{X}]$. For polynomials $f \in S$, the highest term, the leading coefficient, and the leading monomial of f with respect to $\prec_S$ is denoted by $\text{HT}(f)$, $\text{HC}(f)$, and $\text{HM}(f) = \text{HT}(f)\text{HC}(f)$ as usual.

A *specialization* σ of S is a ring homomorphism $\sigma : R \longrightarrow K'$ into some field K'. Let F be a subset of S and let $\text{ideal}(F)$ denote the ideal generated by F. A finite subset $G \subset S$ is a *comprehensive Gröbner base* for $\text{ideal}(F)$ (with respect to $\prec$), if for all fields K' and all specializations $\sigma : R \longrightarrow K'$ of S, $\sigma(G)$ is a Gröbner base for $\text{ideal}(\sigma(F))$ in $K'[X_1, \ldots, X_n]$ (with respect to $\prec$).

```
 1  List<GenPolynomial<Product<Residue<BigRational>>>> L
     = new ArrayList<GenPolynomial<Product<Residue<BigRational>>>>();
 2  BigRational bf = new BigRational(1);
 3  GenPolynomialRing<BigRational> pfac
     = new GenPolynomialRing<BigRational>(bf,3); // no names given
 4  List<GenPolynomial<BigRational>> F
     = new ArrayList<GenPolynomial<BigRational>>();
 5  GenPolynomial<BigRational> pp;
 6  for ( int i = 0; i < 2; i++) {
 7      pp = pfac.random(5,4,3,0.4f);
 8      F.add(pp);
 9  }
10  Ideal<BigRational> id = new Ideal<BigRational>(pfac,F);
11  id.doGB();
12  ResidueRing<BigRational> rr = new ResidueRing<BigRational>(id);
13  System.out.println("rr = " + rr);
14  ProductRing<Residue<BigRational>> pr
     = new ProductRing<Residue<BigRational>>(rr,4);
15  String[] vars = new String[] { "a", "b" };
16  GenPolynomialRing<Product<Residue<BigRational>>> fac
     = new GenPolynomialRing<Product<Residue<BigRational>>>(pr,2,vars);
17  GenPolynomial<Product<Residue<BigRational>>> p;
18  for ( int i = 0; i < 3; i++) {
19      p = fac.random(2,4,4,0.4f);
20      L.add(p);
21  }
22  System.out.println("L = " + L);
```

Figure 2: Constructing algebraic objects

Comprehensive Gröbner bases can be computed, for example, via Gröbner systems. A *Gröbner system* $\mathcal{G}$ for an ideal(F), $F \subset S$ is a finite set of pairs (γ, G_γ) where γ is a condition and $G_\gamma \subset S$ is a finite set of polynomials, determined by γ. A comprehensive Gröbner base G for an ideal(F) is then obtained as the union of all G_γ, where each γ also determines F. The meaning of 'condition' and 'determined' is explained next. If in S we have ideal(F) = ideal(G) then G is called a *faithful* comprehensive Gröbner base.

A *condition* γ is a finite set $\{z_i(\mathbf{U}) = 0\} \cup \{n_j(\mathbf{U}) \neq 0\}$ of polynomial equations and inequalities. A *coloring* of the ring R by a condition γ associates a color, namely *green*, *red* and *white*, with each polynomial in R. For $a \in R$, a is colored *green* if $a(\mathbf{U}) = 0$ can be deduced from γ, a is colored *red* if $a(\mathbf{U}) \neq 0$ can be deduced from γ, else a is colored *white*. If a is colored c we write color$(a) = c$. The coloring of R is extended to a coloring of S by the coloring of the coefficients. For $p \in S$ we write $p = p_{green} + p_{red} + p_{white}$ with the restriction $p_{green} \succ p_{red} \succ p_{white}$ for $p_c \neq 0$ (for a color c). Note, that we allow p_{white} to contain *green*, *red* and *white* coefficients, but p_{green} and p_{red} may only contain *green* respectively *red* coefficients, if they are not zero. The wording 'deduced' is left unspecified. It may mean simple inspection of the polynomials in γ or the usage of more sophisticated methods, like ideal membership tests.

A polynomial p is said to be *determined* with respect to a condition γ, if $p_{red} \neq 0$ or if

$p_{red} = 0$ and $p_{white} = 0$. A set F of polynomials is said to be determined wrt. γ, if each $p \in F$ is determined wrt. γ. A polynomial p is said to be determined with respect to a set of conditions Γ, if p is determined wrt. each $\gamma \in \Gamma$.

More on the mathematical background can be found in [23, 25, 24], see also [20, 21, 3, 16].

3.1 Class Layout

We turn now to the algorithms for the computation of comprehensive Gröbner bases in JAS. Due to space restrictions, we must assume some knowledge of Java, object oriented programming and JAS [9, 7, 11] in the following.

The overall layout of the implemented classes is shown in figure 3. The computation of comprehensive Gröbner bases in class `ComprehensiveGroebnerBaseSeq` is done via Gröbner systems, class `GroebnerSystem`. Gröbner systems are implemented as lists of colored systems in class `ColoredSystem`. The colored systems consist of a tuple of a condition in class `Condition`, a list of colored polynomials and data structure `OrderedCPairlist` representing the critical pairs to be considered. Class `ColorPolynomial` implements a polynomial colored with respect to a certain condition.

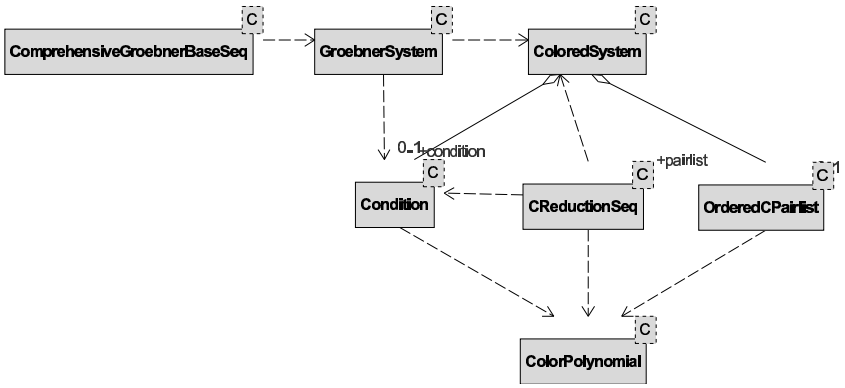


Figure 3: Overview of involved classes

The last class `CReductionSeq` provides methods for parametric reductions relative to conditions and also methods for computing conditions which determine polynomials and sets of polynomials. All classes are parameterized by a type parameter `C` which extends the interface `RingElem<C>`. The implementation is defined for polynomials with polynomial coefficients over a coefficient ring of type `C`, namely `GenPolynomial<GenPolynomial<C>>`.

In the next sub-sections we discuss the functionality of each of the mentioned classes.

3.2 Colored Polynomials and Conditions

Figure 4 shows a class diagram with attributes and methods of the classes `Condition` and `ColorPolynomial`. A condition is defined by a finite set of polynomial equations, polynomials equal to zero $z(\mathbf{U}) = 0$, and a finite set of polynomial inequalities, polynomials not equal to zero $n(\mathbf{U}) \neq 0$. A condition then 'colors' the coefficients of a parametric polynomial in the following way: if a coefficient is contained in the 'equals zero' set, it is colored *green*, if a coefficient is contained in the 'not equals zero' set, it is colored *red*. In case a coefficient is not contained in one of these sets, it is colored *white*. Before we

discuss the implementation of these sets, we first explain the rest of the functionality and the colored polynomials.

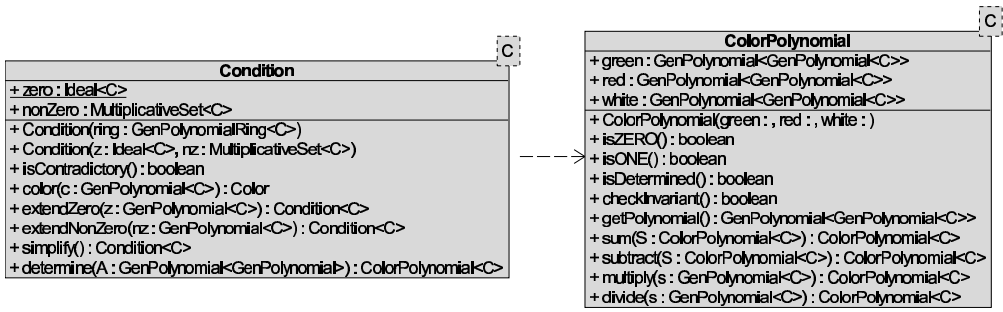


Figure 4: Conditions and colored polynomials

The class `Condition` provides the method `color()` to deduce if a given (parametric) coefficient is zero or not with respect to this condition. The method `determine()` takes a parametric polynomial as input an returns a colored polynomial with respect to this condition. The methods `extendZero()` and `extendNonZero()` add a (parametric) coefficient to the set of zero, respectively the set of non-zero, polynomial equations.

A colored polynomial `ColorPolynomial` consists of these three colored parts determined by a condition, with the following restriction on the ordering on the terms. A non-zero green part `green` is greater with respect to the term order of the main variables than a non-zero red part `red`, which is greater than a non-zero white part `white`. In case, one or more of these parts are zero the restriction holds on the remaining non-zero parts. The method `checkInvariant()` provides a test, if these restrictions are fulfilled. The method `isDetermined()` tests if the red part is non-zero or the white part is also zero. Methods `isZERO()` and `isONE()` ignore the green part in performing the respective test. The `getPolynomial()` method returns the sum of all colored parts. The methods `sum()` and `subtract()` compute a colored polynomial which consists of the sum (difference) of the green parts, a zero red part, and a white part computed from the sum (difference) of the given red and white parts. The methods `multiply()` and `divide()` compute a colored polynomial with each colored part multiplied (divided) by a coefficient.

We now turn to the implementation of the sets of equations and inequalities defining a condition. First we do not store the equations them-self, but only the respective polynomials. The implementation is partially inspired by the implementation in [12] which is based on the implementation of [19].

The test if a polynomial is zero, by inspecting a list of polynomials, is not very efficient. For example, the test polynomial might be a linear combination of some polynomials in the list (in other words, it lies in the ideal generated by the polynomials in the list), a fact which is not detected by just inspecting the list. So we replace the list of polynomials by the ideal generated by the list. Then the test if a polynomial is a linear combination of the polynomials is replaced by an ideal membership test. This test can be performed via a normal form computation modulo a Gröbner base of the ideal generated by the polynomials. This functionality is provided by the class `Ideal`. Its method `contains()` lazily computes a Gröbner base if it is required for the ideal membership test. Further we add the square-free part of the polynomials that are put into the ideal, since the test only requires a radical membership test.

Similarly, the test if a polynomial is non-zero can be improved. Instead of just inspecting the list of polynomials if the given polynomial is contained, we can check if the given polynomial is some product of the polynomials in the list. This is done by computing quotients and remainders with respect to polynomials in the list as long as the remainders are zero. If a quotient is constant, the given polynomial was a product of other non-zero polynomials. This algorithms are implemented in class `MultiplicativeSet`. In subclasses further optimizations are implemented, for example making the polynomials in the set co-prime `MultiplicativeSetCoPrime`, co-prime and square-free `MultiplicativeSetSquarefree` or irreducible `MultiplicativeSetFactors`. The irreducible factors version relies on the new factorization package, which is not yet in a final state. The default is to use squarefree and co-prime multiplicative sets which are also not too expensive to compute.

The methods `extendZero()` and `extendNonZero()` of `Condition` use the tests just described to avoid adding unnecessary polynomials and to add only maximally reduced polynomials to the respective sets. The methods further try to simplify the condition with method `simplify()` and perform checks for contradictions and return `null` as condition in such a case. Contradictions can show up during the extension operations, as a polynomial in the non-zero list might be contained in the extended ideal generated by the zero polynomials. Similarly a polynomial in the zero polynomial ideal could be a product of polynomials in the extended non-zero polynomials set, again a contradiction. In particular the ideal of zero polynomials might contain 1 at some extension operation. Such contradictory conditions can then be given special treatment in the main part of the algorithm.

3.3 Parametric Reductions and Colored Systems

Class `CReductionSeq` implements parametric reductions with respect to conditions. The class diagram is shown in figure 5. The methods `isNormalform()` and `normalform()` test if a polynomial is in reduced form with respect to a list of polynomials or compute such a reduced form relative to a condition and a list of polynomials. All polynomials are colored polynomials as described above and must be colored consistently and be determined. `isNormalform()` checks if a term with a red coefficient is divisible by a red head term of a polynomial in the list.

The computation of the normal form proceeds by inspecting the first non-green term (of the main variables) in the polynomial to be reduced. If it is actually colored green with respect to the condition, then it is put to the green terms of the result polynomial. If it is colored red or white, the term is reduced with respect to a suitable polynomial in the list. If no such polynomial is found, the process ends for top reduction. For non top-reduction the term is put to the result polynomial and the process continues with the next term. Method `SPolynomial()` computes the S-polynomial of two determined polynomials.

The other methods in class `CReductionSeq` implement the computation of sets of conditions and a list of determined polynomial lists. The method `determine()` takes a list of parametric polynomials as input `List<GenPolynomial<GenPolynomial<C>>>` and returns a list of colored systems `List<ColoredSystem<C>>` (explained further down). The method first computes a set of conditions for the list of input polynomials with method `caseDistinction()` and then determines the polynomials with a method `determine()` which takes a case distinction as input.

A case distinction (a set of conditions) is represented by a list of `Condition` objects. The conditions are constructed in a way, such that every polynomial will have a red head term (or the white part is zero). In the construction of the condition, each (parametric)

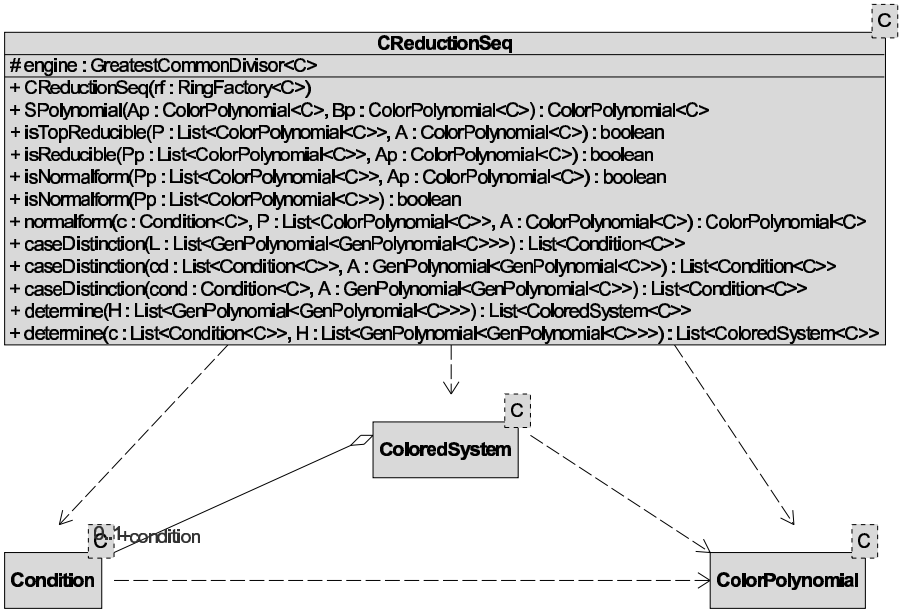


Figure 5: Parametric reduction

coefficient of the given polynomial is checked if it is already colored red or green relative to a given condition. If this is not the case, i.e. the coefficient is colored white, the condition is extended two times. First it is extended by adding the coefficient to the set of non-zero polynomials and then it is extended again by adding the coefficient to the set of zero polynomials (as explained in the previous section). If such a newly computed condition is not contradictory and is not already contained in the list of conditions, it is added to the list of conditions.

The two methods `determine()` take a list of parametric polynomials and return a list of `ColoredSystems`, see figure 6. A `ColoredSystem` is a container for a `Condition`, which determines a list of `ColorPolynomials` and a `OrderedCPairlist`. Besides the pair-list which is explained below, the `ColoredSystem` class provides methods similar to the `ColorPolynomial` class. Namely, there are methods to check for the validity of the term order invariants or to check if the list of polynomials is correctly determined. Further there are methods to extract lists of the green or red coefficients, the essential parts or the parametric polynomials them-selfs. Other methods just return respective parts of the condition.

To construct a list of `ColoredSystems`, method `determine()` with a list of `Condition` parameter, uses a list variant of method `determine()` of class `Condition` to compute a list of colored polynomials from the list of the given parametric polynomials. The condition together with the list of determined colored polynomials are then the building parts for the `ColoredSystem` container. The `determine()` method without a `Condition` parameter, first constructs a set of conditions and then constructs a colored system for each condition in the case distinction.

Class `OrderedCPairlist` implements a data structure for the critical pairs to be considered during the curse of the Buchberger algorithm. It encapsulates pair selection strategies and book keeping for criteria to avoid critical pairs.

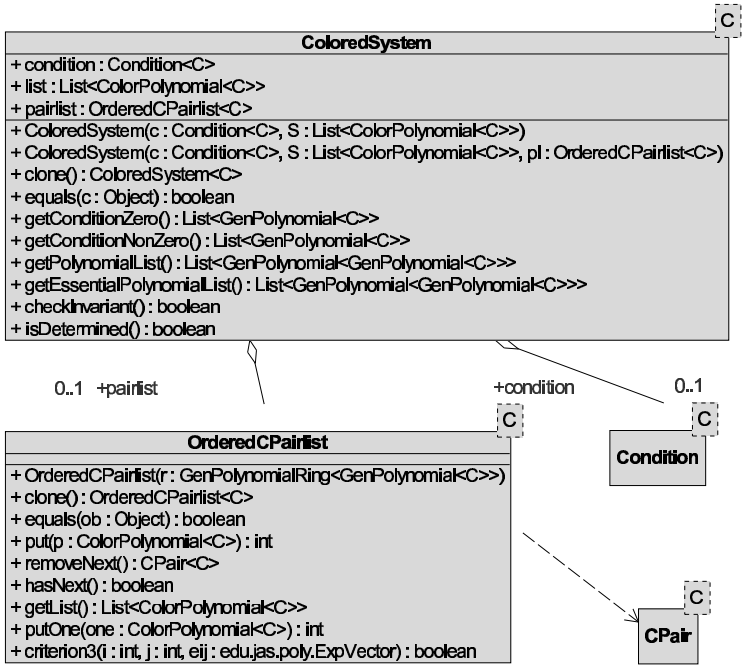


Figure 6: Colored systems and critical pair lists

3.4 Gröbner Systems and Comprehensive Gröbner Bases

A `GroebnerSystem` is a container for a list of `ColoredSystems`, see figure 7. Like class `ColoredSystem`, it has a method `isDetermined()` to test if all contained colored systems are determined and a method `checkInvariant()` to check all invariants of all contained colored polynomials. The method `getConditions()` extracts a list of all `Conditions` from all `ColoredSystems` and stores them for later access in attribute `conds`. The Method `getCGB()` extracts a list of all parametric polynomials as a union of all parametric polynomials from all colored systems.

The computation of comprehensive Gröbner bases via Gröbner systems is implemented in class `ComprehensiveGroebnerBaseSeq`. This class has methods to test if a given list of parametric polynomials is a comprehensive Gröbner base (method `isGB()`) and to test if a given list of colored systems is a Gröbner system (method `isGBsys()`). Both methods are over-loaded to allow also `GroebnerSystems` as parameters and perform the respective checks. Internally there exist two tests, if a list of parametric polynomials is a comprehensive Gröbner base. `isGBcol()` determines the given list of polynomials and calls method `isGBsys()` on the list of `ColoredSystems`. The second test `isGBsubst()` also determines the given list of polynomials but then maps the polynomials to each residue class modulo the zero polynomial ideal contained in the `Condition` and test if it is a Gröbner base over these coefficient rings. That is, we transform the polynomials from

`GenPolynomial<GenPolynomial<C>>` to `GenPolynomial<Residue<C>>`

and use method `isGB()` from implementation `GroebnerBasePseudoSeq` for the test, that is, we map $K[U_1, \ldots, U_m][X_1, \ldots, X_n] \longrightarrow K[U_1, \ldots, U_m]_{/\mathrm{ideal}(Z_i)}[X_1, \ldots, X_n]$, where Z_i is

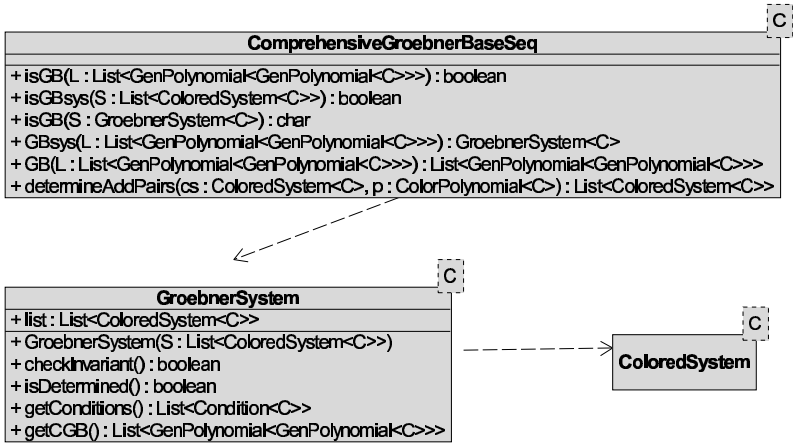


Figure 7: Gröbner systems and comprehensive Gröbner bases

the set of polynomials to be treated as zero in condition i. As one last test a random ideal in the coefficient polynomial ring is generated (an ideal generated by random polynomials) and the test is performed modulo this random ideal. Note, that the ideal(Z_i) of the zero conditions might not be a prime ideal. However, the head terms of the polynomials have moreover been colored red by the multiplicative set of non-zero conditions. And, if the condition is not a contradiction, it is guaranteed that they miss all prime ideals which contain the residue class ideal.

To compute a faithful comprehensive Gröbner base, the method `GB()` first computes a `GroebnerSystem` and then extracts the comprehensive Gröbner base with `getCGB()`.

The main work is performed in method `GBsys()`, which takes a list of parametric polynomials as input an returns a `GroebnerSystem` container. In this method, first the method `determine()` of the parametric reduction engine is used to construct a list of determined colored systems. Each `ColoredSystem` is then augmented by a pair list `OrderedCPairlist` containing all critical pairs of the colored polynomials of it. For each `ColoredSystem`, an inner loop iterates over all critical pairs of this system. For each critical pair a parametric S-polynomial and a parametric normalform of it with respect to the list of colored polynomials is computed. If the normal-form polynomial is non-zero, the condition is refined so that it becomes determined with method `determineAddPairs()`. This method also adds new pairs to the critical pair list if required. The method returns a list of `ColoredSystems`, consisting of successors of the actual condition, an updated list of colored polynomials and an updated list of critical pairs. This list of new `ColoredSystems` is then merged with the existing list of `ColoredSystems` and the actual `ColoredSystem` is replaced by a suitable new system. In this way, a depth first search for a `ColoredSystem` with empty pair list is performed. If all critical pairs of the actual `ColoredSystem` are done, it is moved to the result list and the next colored system is taken. By the termination argument for the computation of Gröbner systems only finitely many new colored systems are added at each step and for each colored system only finitely many new critical pairs are generated. So by Königs tree lemma combined with Dickson's lemma the two interleaved loops eventually terminate. Upon termination all critical pairs in a colored system have been processed, so the polynomials form a Gröbner base relative to the given condition. Since the conditions of the colored system cover the empty condition, the list of colored systems form a Gröbner

system for the list of given input polynomials.

4 Examples and Gröbner bases over regular rings

In this section we report on some performance measurements and relations to Gröbner bases for regular rings. We first show the performance on the examples from Raksanyi and Hawes2, see [2]. The examples are contained in the `examples` directory of [11].

Table 1: Gröbner system for Raksanyi and Hawes examples

example	MAS time	conditions	JAS time	conditions
Raksanyi, S, Gr	40	3	520 / 229 / 190	5
Raksanyi, Lr	not impl.	–	344 / 134 / 94	3
Raksanyi, L	5630	22	511 / 225 / 175	4
Raksanyi, G	30	3	337 / 147 / 99	3
Hawes2, G	> 20 min	–	1119 / 603 / 578	5

time in milliseconds, Term order: G = graded, L = lexicographical, S = Gr = reverse graded, Lr = reverse lexicographical, timings in slashes are for subsequent runs.

In table 1 we compared the computation with MAS [12] on the same computer. For JAS we compute the same Gröbner system three times in the same instance of a Java virtual machine. The time for the second and third run is separated by a slash. We see, that for the first run there is considerable time spend in JVM code profiling and just-in-time compilation. In subsequent runs we then see performance improvements. In most cases, the computing times for the third run are less than half to one third of the computing for the first run. We see that for small examples the MAS code runs faster, but for bigger examples the JAS code runs faster.

Table 2: Gröbner system for Nabeshima examples

example	from [16]	cond	JAS, AMD, L	cond	JAS, AMD, G	cond
F_1	31	4	285 / 151 / 97	7	270 / 142 / 99	7
F_2	93	6	2299 / 1765 / 1664	12	509 / 281 / 165	10
F_3	2203	22	1186 / 720 / 660	29	1199 / 967 / 681	29
F_4	234	15	1231 / 722 / 674	34	1365 / 845 / 751	34
F_5	109	6	359 / 184 / 126	11	367 / 187 / 125	8
F_6	359	17	95 / 43 / 34	4	90 / 42 / 34	4
F_7	375	7	392 / 194 / 117	6	424 / 242 / 128	6
F_8	133200	458	2548 / 1856 / 1788	32	4883 / 4043 / 3664	32

time in milliseconds, Term order: G = graded, L = lexicographical, cond = number of conditions, timings in slashes are for subsequent runs.

In table 2 we present the JAS computation times of the examples from Nabeshima. The timings of Nabeshima are the original timings from the article [16]. These timings are measured on a Pentium M running at 1.73 GHz. The computing times for JAS are in milliseconds on one (older) AMD 2.1 GHz CPU, with JDK 1.6 and 32-bit server VM. So the timings are not directly comparable, but the CPU influence should not be more than a factor of two. Nevertheless we can draw the conclusion, that the CPU or the system software (C, Risa/Asir versus Java, JAS) is qualitatively in the same speed region

and the timing differences seem to be mainly caused by different algorithms. That is, the mathematical optimization to produce a minimal number of conditions, is more important than the relative CPU speed.

Table 3: Gröbner system for Montes examples

example	JAS time	conditions	DISPGB time	conditions
11.1, L	777 / 308 / 327	23	8800	6
11.2, L	490 / 246 / 143	10	5200	6
11.3, L	1013 / 600 / 516	9	115900	7
11.4, L	371939 / 359274 / 355794	7	33000	7
5.1 simpl., L	248 / 95 / 86	3	8400	4

time in milliseconds, Term order: G = graded, L = lexicographical, timings in slashes are for subsequent runs. DISPGB times from [14].

In table 3 we present the JAS computation times of the examples from Montes. The timings of Montes are the original timings from [14]. As in the examples above we conclude, that the different algorithms are most important for the different computing times. As it is not the primary focus of this paper to compare different algorithmic details the timings show that our object oriented approach with Java is not slower than other approaches.

As pointed out in [17, 18, 24] there is some strong relation between comprehensive Gröbner bases and Göbner bases over (von Neumann) regular rings [22]. Since we also have Gröbner bases over (finite) regular rings implemented in JAS, we can check, for example, if a comprehensive Gröbner base is indeed also a Gröbner base over a suitable regular ring.

As with method `isGBsubst()` we take a list of colored polynomials from `GroebnerSystem` (which could be a Gröbner system). From the condition of each colored system we construct a residue class ring modulo the ideal generated from the condition zero polynomials. The (finite) product of these residue class rings are then used as coefficient ring for a polynomial ring with type

```
GenPolynomialRing<Product<Residue<C>>>
```

or in mathematical notation $\left(\prod_{i=1}^{k} K[U_1,\ldots,U_m]_{/\mathrm{ideal}(Z_i)}\right)[X_1,\ldots,X_n]$, where k is the number of colored systems. See section 2 for the construction of the required polynomial rings. Then the union of the parametric polynomials from the colored polynomials is mapped to this polynomial ring. The described conversion is implemented by the (static) method `toProductRes()` of class `PolyUtilApp`. The boolean closure of such a list of polynomials is constructed by method `booleanClosure()` of class `RReductionSeq`. The computation of Gröbner bases for a regular ring is provided via class `RGroebnerBasePseudoSeq<C>` and method `GB()`. The test for a Gröbner base is implemented by method `isGB()`. So if we start with a boolean closed set derived from a comprehensive Gröbner system, the method `isGB()` of the regular coefficient ring Gröbner base will return `true`. An example is contained in the jython file `examples/raksanyi_cr.py` in [11]. Note, as mentioned above, the ideal(Z_i) of the zero conditions might not be a prime ideal. However the head terms of the polynomials have also been colored red by the multiplicative set of non-zero conditions, so it is guaranteed that they miss all prime ideals which contain the residue class ideal, if the condition is not contradictory.

5 Conclusions

We have presented an implementation of the algorithms for computing comprehensive Gröbner bases in a Java computer algebra system (JAS). We provide and utilize all necessary algebraic structures occurring in the comprehensive Gröbner bases algorithm, such as parametric polynomials, colored polynomials, conditions or colored systems. A condition is implemented as an ideal, with normal Gröbner base computations to decide ideal membership and a multiplicative set which is targeted to produce polynomials of minimal degrees using square-free decomposition.

The computing times for our object oriented approach using Java are at least as fast as the times of other implementations. Differences in the computing times are from different mathematical details, which have not been the primary focus of investigation in this paper. With our explicit algebraic types approach we showed how to transform a comprehensive Gröbner system to a polynomial ring over a (commutative, finite, von Neuman) regular coefficient ring and test for Gröbner bases in such polynomial rings.

In the future we will finish the implementation of multivariate polynomial factorization and use it in the handling of the conditions. Further we plan to implement comprehensive Gröbner bases for parametric solvable polynomial rings [13]. There are also many opportunities for utilizing parallelism, see [3] and [10] for a start.

Acknowledgments

I thank Thomas Becker for discussions on the implementation of a polynomial template library and Raphael Jolly for the discussions on the generic type system suitable for a computer algebra system. Thanks also to Markus Aleksy and Hans-Günther Kruse for encouraging and supporting this work. JAS itself has improved by requirements from Axel Kramer and Brandon Barker and by valuable feedback from other colleagues, in particular by Dongming Wang, Thomas Sturm and Wolfgang K. Seiler, to name a few. Finally I thank the anonymous referees for suggestions to improve the paper.

References and Notes

[1] A. Dolzmann and Th. Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bull.*, 31(2):2–9, 1997.

[2] Hans-Gert Gräbe. The SymbolicData project. Technical report, see http://www.symbolicdata.org, accessed 2007, June, 2000–2006.

[3] Shutaro Inoue and Yosuke Sato. On the parallel computation of comprehensive Gröbner systems. In *Proc. PASCO'07*, pages 99–101, 2007.

[4] D. Kapur. An approach for solving systems of parametric polynomial equations. In *Principles and Practices of Constraint Programming*, pages 217–244. MIT Press, Cambridge, Mass., 1995.

[5] H. Kredel. On the Design of a Java Computer Algebra System. In *Proc. PPPJ 2006*, pages 143–152. University of Mannheim, 2006.

[6] H. Kredel. Evaluation of a Java Computer Algebra System. In *Proceedings ASCM 2007*, pages 59–62. National University of Singapore, 2007.

[7] H. Kredel. Evaluation of a Java computer algebra system. *Lecture Notes in Computer Science, Springer Berlin / Heidelberg*, 5081:121–138, 2008.

[8] H. Kredel. Multivariate greatest common divisors in the Java Computer Algebra System. In *Proc. Automated Deduction in Geometry (ADG)*, pages 41–61. East China Normal University, Shanghai, 2008.

[9] H. Kredel. On a Java Computer Algebra System, its performance and applications. *Science of Computer Programming*, 70(2-3):185–207, 2008.

[10] H. Kredel. Distributed parallel Gröbner base computation. In *Proc. Workshop on Engineering Complex Distributed Systems (ECDS) at CISIS 2009*, pages on CD–ROM. University of Fukuoka, Japan, 2009.

[11] H. Kredel. The Java algebra system (JAS). Technical report, http://krum.rz.uni-mannheim.de/jas/, since 2000.

[12] H. Kredel and M. Pesch. *MAS: The Modula-2 Algebra System*, pages 421–428. in Computer Algebra Handbook, Springer, 2003.

[13] Heinz Kredel. *Solvable Polynomial Rings*. Dissertation, Universität Passau, 1993.

[14] A. Montes. An new algorithm for discussing Gröbner basis with parameters. *J. Symb. Comput.*, 33(1-2):183–208, 2002.

[15] A. Montes and M. Manubens. Improving DISPGB algorithm using the discriminant ideal. *J. Symb. Comput.*, 41:1245–1263, 2006.

[16] Katsusuke Nabeshima. A speed-up of the algorithm for computing comprehensive Gröbner systems. In *Proc. ISSAC 2007*, pages 299–306, 2007.

[17] Y. Sato and A. Suzuki. Gröbner bases in polynomial rings over von Neumann regular rings – their applications. In *Proceedings ASCM 2000*, pages 59–62. World Scientific Publications, Lecture Notes Series on Computing, 8, 2000.

[18] Yosuke Sato, Akira Nagai, and Shutaro Inoue. On the computation of elimination ideals of boolean polynomial rings. In *ASCM*, pages 334–348, 2007.

[19] Elke Schönfeld. Parametrische Gröbnerbasen im Computer Algebra System ALDES / SAC-2. Diplomarbeit, Universität Passau, Passau, 1991.

[20] Akira Suzuki and Yosuke Sato. An alternative approach to comprehensive Gröbner bases. *J. Symb. Comput.*, 36(3-4):649–667, 2003.

[21] Akira Suzuki and Yosuke Sato. A simple algorithm to compute comprehensive Gröbner bases using Gröbner bases. In *Proc. ISSAC 2006*, pages 326–331, 2006.

[22] V. Weispfenning. Gröbner bases for polynomial ideals over commutative regular rings. In H.J. Davenport, editor, *Proc. ISSAC'87*, pages 336–347. Springer Verlag, 1987.

[23] V. Weispfenning. Comprehensive Gröbner bases. *J. Symb. Comp.*, 14(1):1–29, 1992.

[24] V. Weispfenning. Comprehensive Gröbner bases and regular rings. *J. Symb. Comput.*, 41:285–296, 2006.

[25] Volker Weispfenning. Canonical comprehensive Gröbner bases. In *ISSAC 2002*, pages 270–276. ACM, 2002.

Computing Monodromy Groups defined by Plane Algebraic Curves by using Extended Hensel Construction

Takaki kubo*

* Graduate School of Pure and Applied Sciences, University of Tsukuba,
Tsukuba, Ibaraki 305-8571, Japan,

Abstract

A symbolic-numeric method is presented for the computed of monodromy groups of plane algebraic curves, by using the extended Hensel construction. For Puiseux expansion to be used for analytic continuation, we employ an algorithm developed by Inaba, Shiihara and Sasaki based on the extended Hensel construction combined with Smith's theorem. The extended Hensel construction allows up to compute all the Puiseux-series roots of a given polynomial efficiently and stably, even with floating-point numbers. Hence, our method can obtain the local monodromies of plane algebraic curves containing floating-point coefficients around each critical point. A computational example is given, explaining a control of the jump of Puiseux-series roots. In addition, we show the calculated result of a monodromy group generated by local monodromies.

1 Introduction

If we trace analytically a convergent power series along a path in the complex plane, we may find a multiple-valued function. The multiple-valuedness is called monodromy. This paper presents an application of the extended Hensel construction (EHC in short) for computing monodromy groups defined by plane algebraic curves. In [DH01], Deconinck and van Hoeij proposed a method for computing Riemann matrices of algebraic curves. The computation of monodromy groups is one step of the computes. The method is based on analytic continuation of algebraic functions along paths in the complex plane. The analytic continuation is a very impotant operation in mathematics: for the determination of Riemann matrices of algebraic curves, the computation of Abel's map, and so on [DH01, DP08]. Nearly a decade ago, Sasaki et al. [SS96, IS04] proposed an algorithm for analytic continuation based on the EHC combined with Smith's theorem. Furthermore, Inaba and Sasaki [IS04] proposed an enhanced method by using a bound for the smallest root [TS00]. In this paper, we apply the method based on Smith's theorem to analytic continuation.

Deconinck and van Hoeij [DH01] use first order approximations in their analytic continuation process, and avoid critical points as much as possible. At the same time, Poteaux [Pot07] made improvements to the method for computing monodromy groups by using an enhanced Newton-Puiseux method. Hence, we can continue analytically Puiseux-series roots above critical points of algebraic curves with floating-point numbers. In order to choose a path of analytic continuation, Poteaux use Euclidean minimal spanning tree. Additionally, Poteaux explained the truncation orders of Newton-Puiseux algorithm. In this paper, we employ the EHC to compute the Puiseux series. The EHC has been invented by Sasaki and Kako [SK99] so as to solve multivariate algebraic equations in series form. So far, the EHC applied to an approximate factorization, an analytic factorization, and so on, of polynomial of more than two variables [Ina05, Iwa03]. Note that series obtained by the EHC is essentially different series from multivariate Puiseux series.

In this paper, we investigate an application of the EHC to compute monodromy groups of plane algebraic curves. Using the EHC, it is possible to compute all Puiseux-series roots concurrently, like Durand-Kerner method in numerical computation. Furthermore,

* *E-mail address*: kubo@math.tsukuba.ac.jp

the EHC is essentially a expansion method at branch points and quite stable [SY98] if performed with floating-point numbers (conventional Newton-Puiseux's method is very unstable). Therefore, our method is able to perform analytic continuation with floating-point numbers. Finaly, we can determine monodromy groups of plane algebraic curves.

2 Preliminaries

In this section, we survey briefly calculation of the monodromy group of a plane algebraic curve. For details, see [DH01, Pot07, Mir95]. Let $\mathbb{K}$ be a subfield of the complex number field $\mathbb{C}$, $\bar{\mathbb{K}}$ be its algebraic closure in $\mathbb{C}$ and consider a plane algebraic curve defined over $\mathbb{C}$,

$$\mathcal{C} = \left\{(x, y) \in \mathbb{C}^2 | F(x, y) = 0\right\}, \tag{2.1}$$

where $F(x, y) \in \mathbb{K}[x, y]$ is a squarefree polynomial, monic in the variable y, and we write

$$F(x, y) = y^d + f_{d-1}(x)y^{d-1} + \cdots + f_1(x)y + f_0(x), \quad (d \geq 1). \tag{2.2}$$

Here $f_m(x),\ m = 0, \ldots, d-1$, are polynomials in x. Let $f_m(x) = \sum_n f_{nm}x^n$, where the coefficients f_{nm} are complex numbers. d is the degree of $F(x, y)$ with respect to y. An irreducible polynomial (2.2) defines a plane algebraic curve (2.1) along with a ramified covering $\pi : \mathcal{C} \to \mathbb{C}$ of the complex plane and a compact Riemann surface Γ canonically associated to the curve $\mathcal{C}$. We want to compute the monodromy group defined by this curve. A complex number c such that the univariate polynomial $F(c, y)$ has multiple roots is called a critical point. Critical points are finite since they are precisely the roots of the discriminant of F in y. We denote them by $c_1, \ldots, c_p$, and the set of critical points by $\mathcal{S}$. A point $a \in \mathbb{C}$ is called regular, if the equation

$$F(a, y) = y^d + f_{d-1}(a)y^{d-1} + \cdots + f_1(a)y + f_0(a) = 0 \tag{2.3}$$

has d distinct roots $\{\alpha_1, \ldots, \alpha_d\}$. The set $\{\alpha_1, \ldots, \alpha_d\}$ forms a fiber at a of the covering, denote them by $\pi^{-1}(a)$. By the implicit function theorem, there exist d analytic functions $y_1(x), \ldots, y_d(x)$ such that $\{y_i(a)\} = \pi^{-1}(a)$ and $F(x, y_i(x)) = 0$ in a neighborhood of a for all i. Consider a path $\gamma : [0, 1] \to \mathbb{C} \setminus \mathcal{S}$ which is a loop in the x-plane starting and ending at a that does not meet any critical point. When τ approaches 1, $\gamma(\tau)$ approaches a so that the values of the continuations $\{y_1(\gamma(\tau)), \ldots, y_d(\gamma(\tau))\}$ tend to the fiber $\pi^{-1}(a)$. Choose one analytic solution of equation (2.2) above $\gamma(0)$. One can continue analytically this solution along γ, yielding an analytic function defined in a neighborhood of $\gamma(1)$ which is still a solution of the equation. Now take γ to be a loop with base point a: then analytic continuation along γ (or any other path homotopic to γ) induces a permutation σ of $\{1, \ldots, d\}$ of the fiber $\pi^{-1}(a)$.

$$y_i(\gamma(\tau)) \to y_{\sigma(i)} = y_{\sigma(i)}(a). \tag{2.4}$$

The action of the fundamental group $\pi_1(\mathbb{C} \setminus \mathcal{S}, a)$ thus defined on $\pi^{-1}(a)$ is called the monodromy of $\mathcal{C}$ with base point a. Up to conjugate, it does not depend on the choice

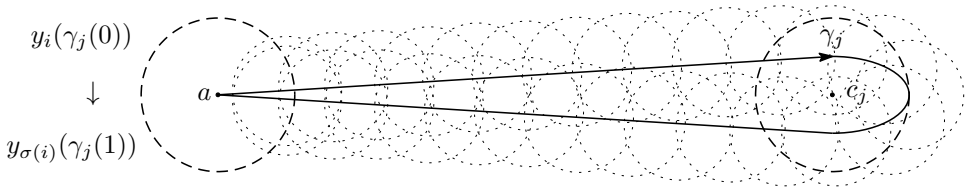


Figure 1: Analytic continuation along the path γ_j around the critical point c_j.

of the base point. The monodromy action of a loop enclosing exactly one critical point c_j is called the local monodromy around c_j. If the permutation σ_j about c_j is not the identity, c_j is called a branch point. The whole monodromy may be represented by the local monodromy around each critical point with respect to a common base point. This is the expected output of an algorithm for monodromy computation.

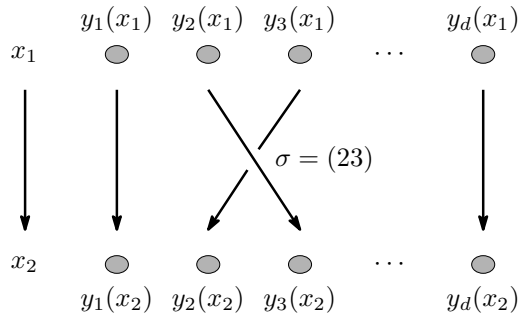


Figure 2: Permutation σ of Puiseux-series roots at x_1 and x_2.

3 Analytic continuation based on the EHC

Many methods for monodromy computation are based upon its definition by means of analytic continuation. Poteaux [Pot07] use an enhanced Newton-Puiseux method to analytic continuation of algebraic function above path in the complex plane. The method is possible to expand at a critical point, and connect near a critical point. At the same time, we apply a symbolic-numeric method for analytic continuation based on the EHC [SS96, IS04] to monodromy computation. In a bivariate case, our method is possible to compute Puiseux-series roots (fractional-power series roots) at a critical point likewise. In addition, the EHC is possible to compute not only series expansion of all roots concurrently but also a multivariate case easily [SK99, SI09]. In this paper, we suppose a bivariate case.

The first of all, we consider the EHC (for details, see [SK99, SI09] and section 5). Let $G_1^{(\infty)}, \ldots, G_d^{(\infty)}$ be Hensel factors, and let $\varphi_1(x), \ldots, \varphi_d(x)$ be roots of the bivariate polynomial (2.2) with respect to y :

$$\begin{aligned} F(x,y) &= G_1^{(\infty)}(x,y)\cdots G_d^{(\infty)}(x,y) \\ &= (y-\varphi_1(x))\cdots(y-\varphi_d(x)). \end{aligned} \tag{3.1}$$

At a critical point, some roots have multiplicity. In this case, we apply the EHC to Hensel factors repeatedly [SK99]. Finally, we obtain factors of (3.1) which are linear with respect to y, i.e. Puiseux expantions of $\varphi_1(x), \ldots, \varphi_d(x)$. We denote them by $y_1(x), \ldots, y_d(x)$. In actual calculation, Hensel factors $G_i^{(\infty)}(x,y)$ $(i = 1, \ldots, d)$ are truncated at kth Hensel factors $G_i^{(k)}(x,y)$. Hence, we obtain the kth fractional-power series expansions with conjugate algebraic coefficients in the neighborhood of a critical point c_j, as follows.

$$y_{i,j}^{(k)}(x) = \sum_{\ell=0}^{k} g_{i\ell}\,(x-c_j)^{\ell/e_i}, \quad e_1 + \cdots + e_s = d, \tag{3.2}$$

where the integers e_i are ramifiication indices. Let a be a regular point: then Puiseux expansions at a regular point a are usual power series expansions, the ramification indices are all 1 and the local monodromy is the identity permutation.

For analytic continuation, we find critical points of a polynomial (2.2) by using the following discriminant.

$$R(x) = \mathrm{Resultant}_y\left(F(x,y), \frac{\partial F(x,y)}{\partial y}\right). \tag{3.3}$$

The roots of the discriminant (3.3) are critical points. We calculate the roots of $R(x)$ by a numerical method. Note that not every root of $R(x)$ is a branch point. Hence, for each root c_j of $R(x)$, we must check whether c_j is a branch point or not. We perform this check by expanding $y = y_{i,j}^{(k)}(x)$ at the point $x = c_j$: if the expansion yields a fractional-power series expansions then c_j is a branch point, otherwise c_j is not a branch point.

Let c_j and $c_{\hat{j}}$ be two different critical points, and let $\zeta_{j\hat{j}}$ be a regular point. We consider the simple case in which the circles of convergence of Puiseux-series roots at $x = c_j$ and $x = c_{\hat{j}}$ overlap and the point $x = \zeta_{j\hat{j}}$ is inside the overlapping area. Furthermore, assume that $\zeta_{j\hat{j}}$ is chosen so that $F(\zeta_{j\hat{j}}, y)$ has no multiple root, then $F(\zeta_{j\hat{j}}, y)$ has d different roots which we put $y = \tilde{\alpha}_1, \ldots, \tilde{\alpha}_d$, $\tilde{\alpha}_s \neq \tilde{\alpha}_t$ for any $s \neq t$. Note that the $\tilde{\alpha}_i$ is computed from the EHC at a regular point $\zeta_{j\hat{j}}$. For $\hat{\ell} = j, \hat{j}$, let $y^{(\infty)}_{i,\hat{\ell}}(x)$ $(i = 1, \ldots, d)$ be the roots of $F(x, y)$ expanded into fractional-power series roots at $x = c_{\hat{\ell}}$, then there is the following one-to-one correspondence between each element of $\{\tilde{\alpha}_1, \ldots, \tilde{\alpha}_d\}$ and each element of $\{y^{(\infty)}_{1,\hat{\ell}}(\zeta_{j\hat{j}}), \ldots, y^{(\infty)}_{d,\hat{\ell}}(\zeta_{j\hat{j}})\}$,

$$y^{(\infty)}_{s_i,j}(\zeta_{j\hat{j}}) = \tilde{\alpha}_i = y^{(\infty)}_{t_i,\hat{j}}(\zeta_{j\hat{j}}), \quad i = 1, \ldots, d, \tag{3.4}$$

where $\{s_1, \ldots, s_d\} = \{1, \ldots, d\} = \{t_1, \ldots, t_d, \}$. The $y^{(\infty)}_{i,\hat{\ell}}(x)$ are infinite fractional-power series roots theoretically, but we can handle only truncated at kth fractional-power series

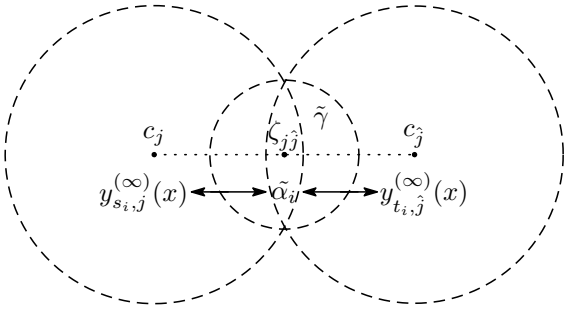


Figure 3: Continuation between $y^{(\infty)}_{s_i,j}(x)$ and $y^{(\infty)}_{t_i,\hat{j}}(x)$

roots $y^{(k)}_{i,\hat{\ell}}(x)$ practcally. Furthermore, the coefficients of $y^{(\infty)}_{i,\hat{\ell}}(x)$ are calculated only approximately. Hence, we cannot calculate $y^{(\infty)}_{i,\hat{\ell}}(\zeta_{j\hat{j}})$ accurately. We can, however, determine the above-mentioned correspondence rigorously by using Smith's theorem [Smi70]. Let $P(y)$ be a monic univariate polynomial in $\mathbb{C}[y]$, and let $z_1, \ldots, z_d$ be d distinct numbers in $\mathbb{C}$. Let d numbers $\rho_1, \ldots, \rho_d$ be defined as follows.

$$\rho_i = \frac{d|P(z_i)|}{\Pi^d_{\hat{i}=1,\neq i}(z_i - z_{\hat{i}})} \quad (i = 1, \ldots, d). \tag{3.5}$$

Let $D_1, \ldots, D_d$ be d discs in the complex plane, such that center$(D_i) = z_i$ and radius$(D_i) = \rho_i$ $(i = 1, \ldots, d)$. Then, the union $D_1 \cup \cdots \cup D_d$ contains all the roots of $D_{\hat{k}+1}, \ldots, D_d$, then the number of roots contained in this union is $\hat{k}$. Let a polynomial $\tilde{P}_\lambda(c_j, c_{\hat{j}}, y)$ be

$$\tilde{P}_\lambda(c_j, c_{\hat{j}}, y) = F(c_j + \lambda(c_{\hat{j}} - c_j), y), \tag{3.6}$$

where λ is a parameter such that $0 \leq |\lambda| \leq 1$. Let $\tilde{D}_i(\lambda)$ $(i = 1, \ldots, d)$ be Smith's discs for $\tilde{P}_\lambda(c_j, c_{\hat{j}}, y)$ with $z_i = \tilde{\alpha}_i$. If $\tilde{D}_1(\lambda), \ldots, \tilde{D}_d(\lambda)$ do not overlap one another for any λ on c_j path connecting 0 and 1, then we have the correspondence $y^{(k)}_{s_i,j}(\zeta_{j\hat{j}}) \longleftrightarrow \tilde{\alpha}_i$. In the same way, the $y^{(k)}_{t_i,\hat{j}}(\zeta_{j\hat{j}})$ is connected to $\tilde{\alpha}_i$. Finally, we get the correspondence $y^{(k)}_{s_i,j}(\zeta_{j\hat{j}}) \longleftrightarrow \tilde{\alpha}_i \longleftrightarrow y^{(k)}_{t_i,\hat{j}}(\zeta_{j\hat{j}})$.

4 Determining the global monodromy

Since we can obtain the local monodromy in any critical point simply by looking at the EHC, all we need to compute the global monodromy is to express local monodromies using

the same base point. And to that end, we need to determine the path γ around a critical point. In this paper, we apply a method in [DH01] to path choice.

Let $V = \{c_0 = a, c_1, \ldots, c_p\}$ be the set of critical points, augmented by the base point a. Note that we choose a so that its real part is less than the real part of each critical point. By this choice, the arguments of $c_j - a$ are between $-\pi/2$ and $\pi/2$. Now paths are chosen for the analytic continuation. The paths chosen are composed of line segments and semi-circles. Let $D(c_j, r(c_j))$ $(j = 1, \ldots, p)$ denote the circle with center $c_j \in V$ and radius $r(c_j)$, and let ξ_j denote the point $c_j - r(c_j)$. The simplest path γ_j around c_j consists of one line segment from a to ξ_j. This is followed by $D(c_j, r(c_j))$, starting at ξ_j. Successively, a line segment is followed from ξ_j, back to a. However, in many cases, this path will intersect one of the circles $D(c_{\hat{j}}, r(c_{\hat{j}}))$, $\hat{j} \neq j$. This indicates that the path comes close to the critical point $c_{\hat{j}}$. To avoid accuracy issues during the analytic continuation, this should be avoided. This is remedied as indicated in figure 4: the path takes a detour along a semi-circles is around $c_{\hat{j}}$. Whether this semi-circles goes above or below $c_{\hat{j}}$ depends on the relative positions of a, c_j and $c_{\hat{j}}$. The semi-circles is chosen such that the new path is deformable to γ_j, without crossing any critical points of the analytic continuation.

This process is iterated, until a path is obtained, which stays at least $r(c_{\hat{j}})$ away from $c_{\hat{j}}$, for $\hat{j} = 1, \ldots, p$. These new paths are homotopic to each $\gamma_1, \ldots, \gamma_p$ in $\mathbb{C}\backslash\mathcal{S}$, and we denote them by $\tilde{\gamma}_1, \ldots, \tilde{\gamma}_p$. In order to calculate the global monodromy, we need to continue analytically the functions $y_i^{(k)}(x)$ $(i = 1, \ldots, d)$ along each path $\tilde{\gamma}_j$. Finally, we obtain the permutations associated to each critical point c_j, and we determine the global monodromy.

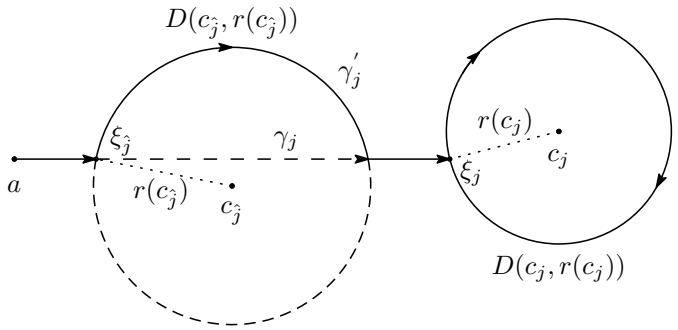


Figure 4: Path homotopic to γ_j

5 The extended Hensel construction

We review the EHC proposed by Sasaki and Kako. For more details, see [SK99, SI09]. For each non-zero term $f_{nm} \cdot x^n y^m$ of a bivariate polynomial (2.2), we plot a dot at the point (n, m) in the two-dimensional Cartesian coordinate system. Let $\mathcal{L}$ be a straight line such that it passes the point $(d, 0)$ as well as another dot plotted and that no dot is plotted below $\mathcal{L}$. The line $\mathcal{L}$ is called Newton's line for $F(x, y)$. Figure 5 shows $\mathcal{L}$. The sum of all the term of a polynomial (2.2), which are plotted on Newton's line is called Newton's polynomial for (2.2). We denote Newton's polynomial by $F_{\mathrm{New}}(x, y)$. Newton's line is uniquely determined by $F(x, y)$. Let Newton's line F_{New} in (e_y, e_x)-

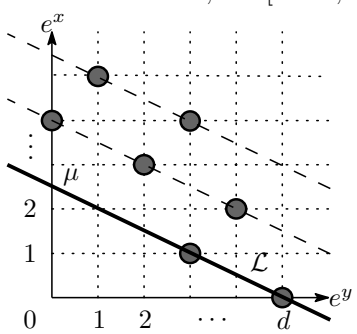


Figure 5: Example of Newton's line $\mathcal{L}$

plane be $e_y/d + e_x/\mu = 1$, where μ is the intersection of $\mathcal{L}$ and e_x-axis. Then, Newton's polynomial consists of some of the terms

$$y^d,\ y^{d-1}x^{\mu/d},\ y^{d-2}x^{2\mu/d}, \ldots, x^{d\mu/d}. \tag{5.1}$$

Let $\hat{\mu}$ and $\hat{d}$ be positive integers such that

$$\hat{\mu}/\hat{d} = \mu/d, \quad \gcd(\hat{\mu}, \hat{d}) = 1. \tag{5.2}$$

Suppose that $F_{\mathrm{New}}(x, y)$ is factorized over $\mathbb{C}$ as

$$\begin{aligned} F_{\mathrm{New}}(x,y) &= G_1^{(0)}(x,y)\cdots G_q^{(0)}(x,y), \quad q \geq 2, \\ \gcd\left(G_{\hat{s}}^{(0)}, G_{\hat{t}}^{(0)}\right) &= 1, \quad \hat{s} \neq \hat{t}. \end{aligned} \tag{5.3}$$

We define an ideal I_k as

$$I_k = \left(y^d x^{(k+0)/\hat{d}}, y^{d-1}x^{(k+\hat{\mu})/\hat{d}}, \ldots, y^0 x^{(k+d\hat{\mu})/\hat{d}}\right), \tag{5.4}$$

and we compute Lagrange's interpolation polynomials. For each value of $m = 0, \ldots, d-1$, there exists only one set of polynomials $\{W_1^{(m)}(x,y), \ldots, W_r^{(m)}(x,y)\}$ satisfying

$$\begin{aligned} W_1^{(m)}(x,y)\frac{F_{\mathrm{New}}(x,y)}{G_1^{(0)}(x,y)} + \cdots + W_q^{(m)}(x,y)\frac{F_{\mathrm{New}}(x,y)}{G_q^{(0)}(x,y)} &= y^m x^{(d-m)\hat{\mu}/\hat{d}}, \\ \deg_y\left(W_{\hat{s}}^{(m)}(x,y)\right) < \deg_y\left(G_{\hat{s}}^{(0)}(x,y)\right), &\quad \hat{s} = 1, \ldots, q. \end{aligned} \tag{5.5}$$

Suppose that $G_1^{(k-1)}, \ldots, G_q^{(k-1)}$ have been calculated. We express $F - G_1^{(k-1)}\cdots G_q^{(k-1)}$ as

$$F - G_1^{(k-1)}\cdots G_q^{(k-1)} \equiv \delta f_{d-1}^{(k)} y^{d-1} x^{(\hat{\mu}/\hat{d})} + \cdots + \delta f_0^{(k)} y^0 x^{d\hat{\mu}/\hat{d}} \pmod{I_{k+1}}. \tag{5.6}$$

Then, we construct $G_{\hat{s}}^{(k)}(x,y)$ by the following formula.

$$G_{\hat{s}}^{(k)}(x,y) = G_{\hat{s}}^{(k-1)}(x,y) + \sum_{m=0}^{d-1} \delta f_m^{(k)}(x) W_{\hat{s}}^{(m)}(x,y), \quad \hat{s} = 1, \ldots, q. \tag{5.7}$$

This construction is called extended Hensel construction. This method gives repeatedly, we finally obtain factors of $F(x,y)$ which are linear with respect to y.

6 A computational experiment

In this section we show the results of application of the methods proposed in the previous sections to example of monodromy group. We consider the following equation.

$$F(x,y) = y^3 - x^7 + 2x^3 y. \tag{6.1}$$

The equation (6.1) come from [DH01]. Let c_0 be a base point, and we calculate critical points of (6.1) from the discriminant (3.3). Results are following.

$$\left\{\begin{array}{rcl} c_0 &=& -1.4483892023\cdots, \\ c_1 &=& -0.3196977699\cdots - 0.98392856357\cdots i, \\ c_2 &=& \ \ 0.8369796279\cdots - 0.60810129478\cdots i, \\ c_3 &=& -1.0345637159\cdots, \\ c_4 &=& \ \ 0.0000000000\cdots, \\ c_5 &=& \ \ 0.8369796279\cdots + 0.60810129478\cdots i, \\ c_6 &=& -0.3196977699\cdots + 0.98392856357\cdots i. \end{array}\right. \tag{6.2}$$

We indicate the above values and each circle $D(c_j, r(c_j))$ $(j = 1, \ldots, 6)$ in figure 6. As a

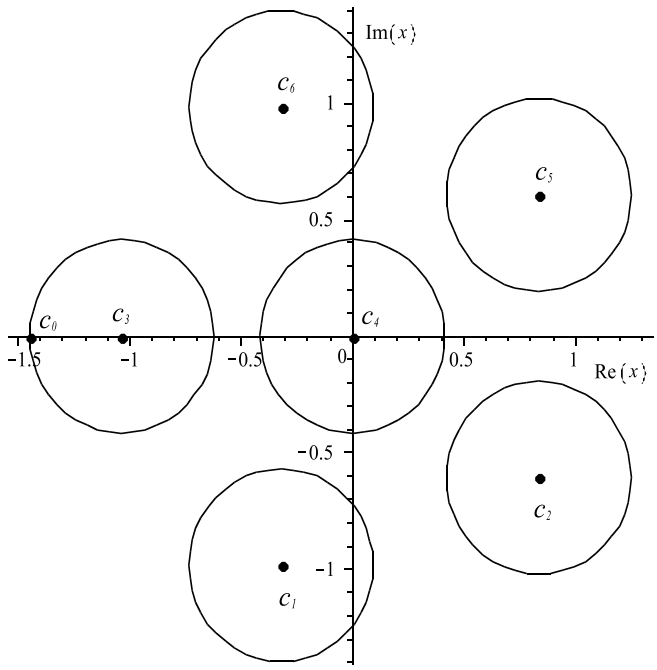


Figure 6: Critical points (6.2)

simple example, we describe Puiseux-series roots of (6.1) about one critical point $c_4 = 0$ by using the EHC, as follows.

$$\begin{cases} y_{1,4}^{(k)}(x) = & \frac{1}{2}x^4 - \frac{1}{16}x^9 + \frac{3}{128}x^{14} + \cdots, \\ y_{2,4}^{(k)}(x) = & \sqrt{2}ix^{3/2} - \frac{1}{4}x^4 + \frac{3}{64}\sqrt{2}ix^{13/2} + \frac{1}{32}x^9 - \frac{105}{8192}\sqrt{2}ix^{23/2} - \frac{3}{256}x^{14} + \cdots, \\ y_{3,4}^{(k)}(x) = & -\sqrt{2}ix^{3/2} - \frac{1}{4}x^4 - \frac{3}{64}\sqrt{2}ix^{13/2} + \frac{1}{32}x^9 + \frac{105}{8192}\sqrt{2}ix^{23/2} - \frac{3}{256}x^{14} + \cdots. \end{cases} \quad (6.3)$$

To make it easier to understand visually, we use the exact value $c_4 = 0$ as the expansion point. In actual calculation, the values of $c_{\hat{j}}$ $(\hat{j} = 0, \ldots, 6)$ are obtained as floating-point numbers. Therefore, the EHC would be a symbolic-numeric approach [SY98, IS07]. At a base point c_0 which is a regular, Taylor-series roots are obtained as following.

$$\begin{cases} y_{1,0}^{(k)}(x) &= -3.20203\cdots + 4.25125\cdots x - 1.68471\cdots x^2 + 0.16816\cdots x^3 + \cdots, \\ y_{2,0}^{(k)}(x) &= 1.60101\cdots - 2.12562\cdots x + 0.84235\cdots x^2 - 0.08408\cdots x^3 + \cdots \\ &\quad + i\left(1.26997\cdots - 3.08354\cdots x + 1.35754\cdots x^2 - 0.46412\cdots x^3 + \cdots\right), \\ y_{3,0}^{(k)}(x) &= 1.60101\cdots - 2.12562\cdots x + 0.84235\cdots x^2 - 0.08408\cdots x^3 + \cdots \\ &\quad - i\left(1.26997\cdots - 3.08354\cdots x + 1.35754\cdots x^2 - 0.46412\cdots x^3 + \cdots\right). \end{cases} \quad (6.4)$$

Let $\tilde{\gamma}_j$ be the paths built in section 4, and let Taylor-series roots $y_{i,0}^{(k)}(x)$ $(i = 1, \ldots, 3)$ be initial functions of analytic continuation along the paths $\tilde{\gamma}_j$. Then, we calculate local monodromies associated to each critical point c_j. Since the $y_{i,j}^{(k)}(x)$ are impossible to continue

over the nearest critical point, we need to connect Taylor-series roots from c_0 to c_3 always first. Furthermore, note that we entail Taylor-series roots at regular points between each critical point. Because Puiseux-series roots at critical points have ramification indices. We plot a phase of the jump of each Puiseux-series root around the critical point c_3 in figure 8 and 9. This behavior does not arise from regular points. Hence, we can control the jump of Puiseux-series roots, and we can continue analytically Puiseux-series roots around each critical point. Let the behavior speak for itself, in figure 10 and 11. In these figures, we evaluate $y_{i,0}^{(20)}(x)$ and $y_{i,3}^{(20)}(x)$ along the circle $x = 0.3\,(\cos(\theta) + i\sin(\theta))$ $(0 \leq \theta \leq 2\pi)$. We can see the jump between $y_{2,3}^{(20)}(x)$ and $y_{3,3}^{(20)}(x)$.

Finally, we obtain permutations related the local monodromy in any critical point as

c_j	c_1	c_2	c_3	c_4	c_5	c_6	∞
σ_j	(13)	(23)	(23)	(13)	(23)	(12)	(132)

Figure 7: Permutations σ_j associated to each critical point c_j.

follows. The collection of all σ_j generates the monodromy group, which is represented here as a subgroup of S_d, the group of permutations of $\{1, \ldots, d\}$. The point $x = \infty$ might also be a branch point. The corresponding permutation σ_∞ does not need to be computed by analytic continuation, since it can be determined from the other σ using the following relation [DH01].

$$\sigma_\infty \circ \sigma_p \circ \sigma_{p-1} \circ \cdots \circ \sigma_1 = 1 \tag{6.5}$$

This states that analytic continuation along a closed path in the extended complex x-plane that encircles all branch points will act as the identity permutation. Such a path is deformable to a point and analytic continuation along this path does not permute the entries of $y_{i,0}(c_0)$.

7 Conclusion

In this paper, we have presented a symbolic-numeric method for computation of monodromy groups of plane algebraic curves based on the EHC. The EHC is essentially a expansion method at critical points and quit stable if performed with floating-point numbers. Hence, we can compute local monodromies of plane algebraic curves with floating-point coefficients. Moreover, the EHC may compute a multivariate case, that is about an algebraic surface. Let $F(x, u_1, \ldots, u_{\hat{q}})$, with $\hat{q} \geq 2$, be an irreducible multivariate polynomial over some subfield $\mathbb{K}$ of the complex numbers. Then we can obtain a multivariate Puiseux-series roots $\varphi_i(u_1, \ldots, u_{\hat{q}})$, $(i = 1, \ldots, d)$, such that

$$\begin{aligned} F(x, u_1, \ldots, u_{\hat{q}}) &= G_1^{(\infty)}(x, u_1, \ldots, u_{\hat{q}}) \cdots G_d^{(\infty)}(x, u_1, \ldots, u_{\hat{q}}) \\ &= (x - \varphi_1(u_1, \ldots, u_{\hat{q}})) \cdots (x - \varphi_d(u_1, \ldots, u_{\hat{q}})). \end{aligned} \tag{7.1}$$

Note that in multivariate case we introduce the total-degree variable $\bar{t}$ by the replacement $u_1 \to \bar{t}u_1, \ldots, u_{\hat{q}} \to \bar{t}u_{\hat{q}}$ in $F(x, u_1, \ldots, u_{\hat{q}})$. Acoordingly, we determine Newton's line $\mathcal{L}$ in $(e_y, e_{\bar{t}})$-plane [SK99]. As a result, the EHC is possible to continue analytically multivariate Puiseux-series roots above critical points. However, in a multivariate case, the treatment of singularities, the region of convergence, fundamental groups, and so on, become a problem. These problems are future tasks.

Acknowledgements

The author expresses his sincere thank to Prof. Tateaki Sasaki for useful discussions of this topic and instructive advice on the extended Hensel construction.

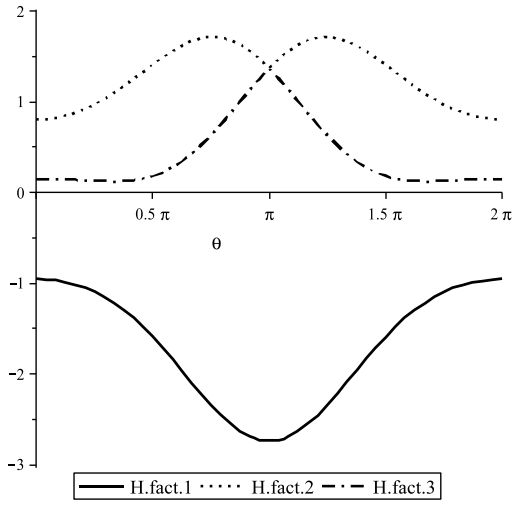


Figure 8: Real part of $y_{i,3}^{(20)}(x)$ around c_3.

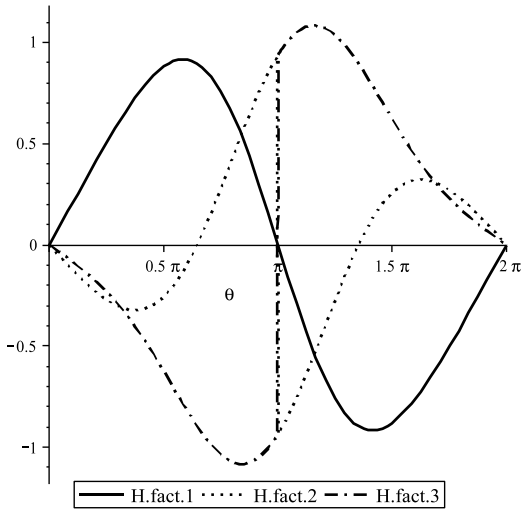


Figure 9: Imaginary part of $y_{i,3}^{(20)}(x)$ around c_3.

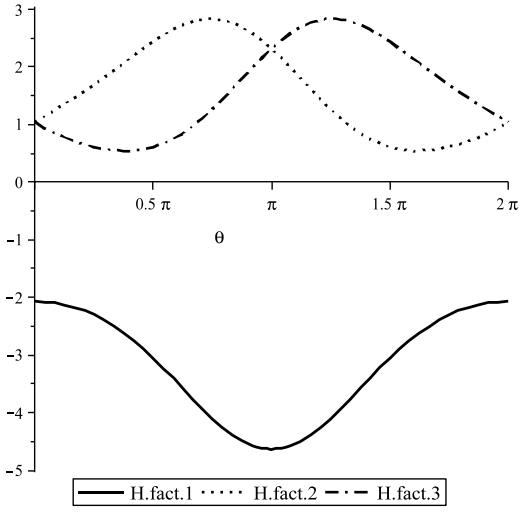


Figure 10: Real part of $y_{i,0}^{(20)}(x)$ around c_0.

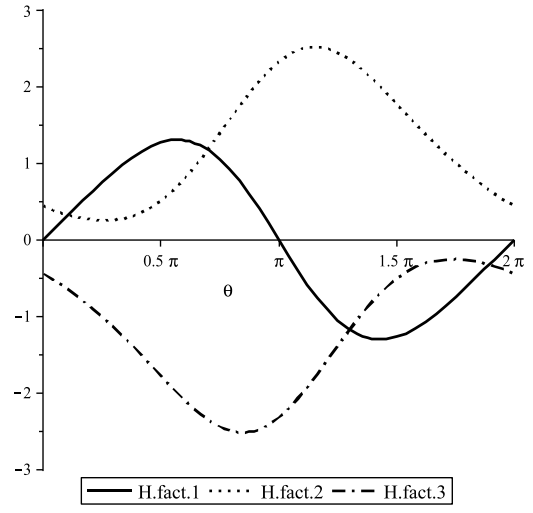


Figure 11: Imaginary part of $y_{i,0}^{(20)}(x)$ around c_0.

References and Notes

[DP08] B. Deconinck and M. S. Pattarson, Computing Abel map, *Physica D*, **237**, 3214–3232, 2008.

[DH01] B. Deconinck and M. van Hoeij, Computing Riemann matrices of algebraic curves, *Physica D*, **152/153**, 28–46, 2001.

[Ina05] D. Inaba, Factorization of multivariate polynomials by extended Hensel construction, *ACM SIGSAM Bulletin*, **39(1)**, 2-14, 2005.

[IS04] D. Inaba, T. Sasaki, Certification of analytic continuation of algebraic function, *Proc. CASC'04 (Computer Algebra in Scientific Computation)*, V. G. Ganzha, E. W. Mayr and E. V. Vorozhtsov (Eds.), Technishe Universität München Press, 249–260, 2004.

[IS07] D. Inaba, T. Sasaki, A numerical study of extended Hensel series, *Proc. SNC'07 (Symbolic-numeric computation)*, J. Versched and S. T. Watt (Eds.), ACM, 103–109, 2007.

[Iwa03] M. Iwami, Analytic factorization of the multivariate polynomial, *Proc. CASC 2003 (Computer Algebra in Scientific Computing)*, V. G. Ganzha, E. W. Mayr and E. V. Vorozhtsov (Eds.), Technishe Universitat Munchen Press, 213–225, 2003.

[Mir95] M. Miranda, *Algebraic curves and Riemann surfaces*, Graduate Studies in Mathematics, American Mathematical Society, Providence, RI, 1995.

[Pot07] A. Poteaux, Computing monodromy groups defined by plane algebraic curves, *Proc. SNC'07 (Symbolic-numeric computation)*, J. Versched and S. T. Watt (Eds.), ACM, 36–45, 2007.

[SI09] T. Sasaki, D. Inaba, Convergence and many-valuedness of hensel seriesnear the expansion point, *Proc. SNC'09 (Symbolic-numeric computation)*, ACM, 159–168, 2009.

[SK93] T. Sasaki and F. Kako, Solving multivariate algebraic equation by Hensel construction, Preprint of Univ. Tsukuba, March, 1993.

[SK99] T. Sasaki and F. Kako, Solving multivariate algebraic equation by Hensel construction, *Japan J. Indus. Appl. Math.*, **16(2)**, 257–285, 1999.

[Smi70] B. T. Smith, Error bounds for zeros of a polynomial based upon Gerschgorin's theorems, *J. ACM*, **17**, 661–674, 1970.

[SS96] K. Shiihara and T. Sasaki, Analytic continuation and Riemann surface determination of algebraic functions by computer, *Japan J. Indust. Appl. Math.*, **13(1)**, 107–116, 1996.

[SY98] T. Sasaki and S. Yamaguchi, An analysis of cancellation error in multivariate Hensel Construction with floating-point number arithmetic, *Proc. ISSAC'98 (Intern'l. Symp. on Symbolic and Algebraic Computation)*, ACM, 1–8, 1998.

[TS00] A. Terui and T. Sasaki, "Approximate zero-points" of real univariate polynomial with large error terms, *IPSJ (Information Processing Society of Japan) J.*, **41**, 974–989, 2000.

A Family of Block Numerical Multistage-Multistep Method with Advanced Step-Points

Ming-Gong Lee* and Rei-Wei Song
mglee@chu.edu.tw*
Department of Applied Mathematics/ College of Engineering
Chung Hua University, Hsinchu, 30012 Taiwan

Abstract: A special class of multistage and multistep integration methods which can obtain r new values simultaneously at each integration step were developed. The block extended backward differentiation formulas also contain extended step-point in the formulas. Their stability regions were sketched, and their regions are either A-stable or $A(\alpha)$-stable. In addition, in a predictor-corrector scheme, their stability interval is one of the largest among some known articles. Applications of the block formulas to numerical solutions of stiff differential equations by Newton's scheme were also studied.

Key Words: Multistage and multistep methods, Extended backward differentiation formulas, A-stable, $A(\alpha)$-stable, Predictor-corrector scheme, Stiff equations

1. Introduction

Numerical solutions for ordinary differential equations (ODEs) have great importance in scientific computation, as they were widely used to model in representing the real world problems [9]. The common methods used to solve ODEs are categorized as one-step (multistage) methods and multistep (one stage) methods, which Runge-Kutta methods represent the former group, and Adams-Bashforth-Molton method represents the later group. Some multistage methods are available in the community. Implicit one-step method has been studied by Stoller and Morrison [9], Butcher [5], and Shampine and Watts [9]. But, here we tried to set up a system with combination of different number of stages and step formulas, in addition, these formulas have good stability region for ordinary differential equations.

In this paper we shall be concerned with the approximate numerical solutions of the stiff initial value problem

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0 \tag{1}$$

on the finite interval $I=[x_0,x_N]$ where $y:[x_0,x_N]\to R^m$ and

$f:[x_0,x_N]\times R^m\to R^m$. Many efficient numerical methods have been proposed, but despite the algorithm developed by Gear [6] more than 3 decades ago which remains one of the most efficient general purpose algorithms. We quoted the following description from [2] to describe the importance of this algorithm. Three of the most important factors contributing to the success of this algorithm, which incorporates backward differentiation formulae of orders 1-6 are:

(1) the relative ease with order and stepsize maybe changable,

(2) the possibility of using higher order, highly stable schemes,

(3) the relatively small amount of computational effort required per step.

Based on this description, same idea of adapting backward differentiation is also implemented, we consider a class with both explicit and implicit multistep and multistage methods for solving ordinary differential equations, and we can call it a block method. For nonstiif ODEs, it can be implemented by a predictor and corrector schemes since it contains explicit and implicit formulas. In addition, if dealing with stiff problems, implicit formulas can be used solely by some Newton-like methods if computing complexity is a concern. The advantage of our method is that it can obtain a block of new values simultaneously which makes it be more competitive. A stepsize adjustment strategies was developed to be implemented in the predictor-corrector scheme.

As for the rest of the paper, we organized the content as the following. In section 2, we will show some of the block multistage/multistep formulas, and their stability regions will be sketched and be given at section 3. In section 4, we implement a variable stepsize strategy for moderate stiff ODEs by Block formulas. Some numerical results for stiff ODEs will be given too. Conclusion is given at section 5.

2. Block Multistage and Multistep Method

Given an s stages and m steps integration formula [1, 7], where

$h^{-1}(\sum_{j=0}^{m}k_j^q y_{i+s-j})$ approximates $y'(x_{i+q})$ of order h^m, $q=1,2,...,s$, and $m\ge s$. Define an s stages and m steps extended backward differentiation scheme to be one taking the general form

$$\begin{cases}\bar{\alpha}_{1,s}y_{n+s}+\bar{\alpha}_{1,s-1}y_{n+s-1}+...+\bar{\alpha}_{1,s-m}y_{n+s-m}=h\bar{\beta}_{1,s-m}f_{n+s-m},\\ \qquad\qquad\vdots\\ \bar{\alpha}_{s,s}y_{n+s}+\bar{\alpha}_{s,s-1}y_{n+s-1}+...+\bar{\alpha}_{s,s-m}y_{n+s-m}=h\bar{\beta}_{s,s-m}f_{n+s}.\end{cases}\tag{2.1}$$

A few formulas are given in the following:

a. For s=m=1, it is exactly the same as the classical BDF formula.

b. For s=2, m=3, explicit and implicit formula are the following:

$$\begin{aligned} &2y_{n+2}-9y_{n+1}+18y_n-11y_{n-1}=6hf(x_{n-1},y_{n-1}) \\ &-y_{n+2}+6y_{n+1}-3y_n-2y_{n-1}=6hf(x_n,y_n). \end{aligned} \tag{2.2}$$

$$\begin{aligned} &2y_{n+2}+3y_{n+1}-6y_n+y_{n-1}=6hf(x_{n+1,}y_{n+1}) \\ &11y_{n+2}-18y_{n+1}+9y_n-2y_{n-1}=6hf(x_{n+2},y_{n+2}) \end{aligned} \tag{2.3}$$

c. For s=3, m=5, explicit and implicit formulas are the following:

$$\begin{aligned} &-12y_{n+3}-75y_{n+2}+200y_{n+1}-300y_n+300y_{n-1}-137y_{n-2}=60hf(x_{n-2},y_{n-2}) \\ &-3y_{n+3}+20y_{n+2}-60y_{n+1}+120y_n-65y_{n-1}-12y_{n-2}=60hf(x_{n-1},y_{n-1}) \\ &2y_{n+3}-15y_{n+2}+60y_{n+1}-20y_n-30y_{n-1}+3y_{n-2}=60hf(x_n,y_n) \end{aligned} \tag{2.4}$$

$$\begin{aligned} &-3y_{n+3}+30y_{n+2}+20y_{n+1}-60y_n+15y_{n-1}-2y_{n-2}=60hf(x_{n+1},y_{n+1}) \\ &12y_{n+3}+65y_{n+2}-120y_{n+1}+60y_n-20y_{n-1}+3y_{n-2}=60hf(x_{n+2},y_{n+2}) \\ &137y_{n+3}-300y_{n+2}+300y_{n+1}-200y_n+75y_{n-1}-12y_{n-2}=60hf(x_{n+3},y_{n+3}) \end{aligned} \tag{2.5}$$

Additional formulas can be derived by the same way, and we will not give detail here. If we let the leading coefficient of y_{n+s} to be one, and additional function evaluation at advanced step-point are equipped into the original formulas, then a new block formulas can be derived, and interestingly, the formulas are similar to the EBD(extended Backward Differentiation Formulas) which J.R. Cash derived in [2]. The derivation will be given in the next section. Some reports show that EBD has great performance in solving stiff ODEs. Also a MEBDF (modified extended backward differentiation formulas) are derived [3] and has great performance in solving stiff ODEs and DAEs. We will investigate if our methods have good performance to stiff ODEs and some DAEs in future research. Our block formulas can be implemented by either Newton iteration with the implicit formulas, such as Formulas 2.3 and 2.6 or by a predictor-corrector scheme, such as Formulas 2.2 and 2.3; 2.5 and 2.6, etc. Some numerical results will be given in section 4.

3. Stability property

The main difficulty associated with stiff equations is that even though the component of the true solutions corresponding to some eigenvalues that may becoming negligible, the restriction on the stepsize imposed by the numerical stability of the method requires that $|h\lambda|$ remain small throughout the range of integration. So a suitable formula for stiff equations would be the one that would not require that $|h\lambda|$ remains small. Dahlquist [5] investigated the special stability problem connected with stiff

equations, he introduced the concept of A-stability, and we quote the definition as the following:

Definition 3.1: The stability region *R* associated with a multistep formula is defined as the set $R = \{h\lambda : \text{A numerical formula applied to } y' = \lambda y,\ y(x_0) = y_0, \text{ with constant stepsize } h > 0, \text{ produce a sequence } (y_n) \text{ satisfying that } y_n \to 0 \text{ as } n \to \infty\}$.

Definition 3.2: A formula is A-stable if the stability region associated with that formula contains the open left half complex place.

Definition 3.3 [5]: A convergent linear multistep method is $A(\alpha)$-stable, for $0 < \alpha < \pi/2$, if $S_\alpha = \{\mu : |\arg(-\mu)| < \alpha,\ \mu \neq 0\}$. A method is A(0)-stable if it is $A(\alpha)$-stable for some (sufficiently small) $\alpha > 0$.

To derive the region of absolute stability, one may consider the model problem for ODEs. We apply formulas 2.3 and 2.6 to $y' = \lambda y$, $y(x_0) = y_0$, and manipulation is skipped here. Let $\mu = \lambda h$, we have the following locus plots for the block formulas:

a. Locus plot of Formula 2.3 (implicit block formula) :

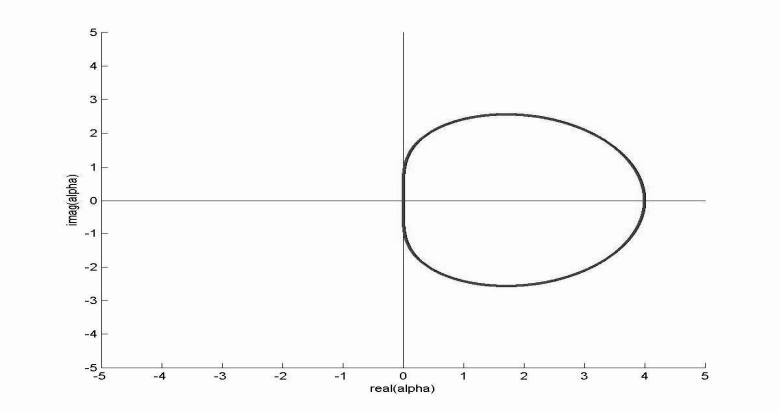


Figure 3.1: Stability region of implicit Formula 2.3
The region is the exterior of the blue line

We notice that in Figures 3.1, the stability region is at the exterior of the blue line and it shows that Formula 2.3 is an A-stable method, which means the regions covers entirely left half plan.

b. Locus plot of Formula 2.5 (implicit block formula):

We notice that in Figures 3.2, the stability region is at the exterior of the blue line and it shows that Formula 2.5 is an $A(\alpha)$-stable method, which means the regions covers almost all the left half plan. The other regions of absolute stability were not sketched and details will be given in the outcoming paper.

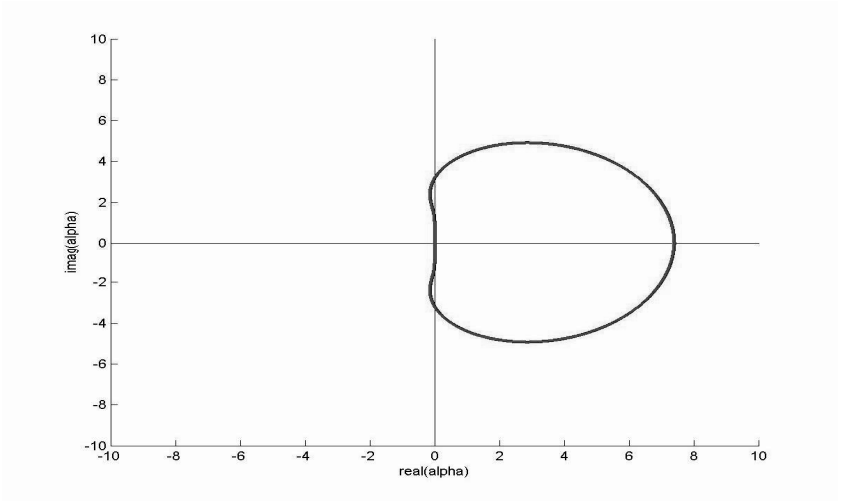


Figure 3.2: Stability region of implicit Formula 2.6
The region is the exterior of the blue line

For estimation of the absolute stability region of the Block Predictor-Corrector scheme, we derive in the following. Let H = $h\lambda$. We could rewrite Formulas 2.2 and 2.3 (or similarly Formulas 2.4 and 2.5) by the following:

$$\begin{aligned} Y_m^p &= AY_{m-1}^c + hBF_{m-1}^c \\ Y_m^c &= A^*Y_{m-1}^c + hB^*F_{m-1}^p \end{aligned} \tag{3.1}$$

For s=2, m=3,

$$\begin{bmatrix} y_{n+1}^p \\ y_{n+2}^p \end{bmatrix} = \begin{bmatrix} 5 & -4 \\ 28 & -27 \end{bmatrix} \begin{bmatrix} y_{n-1}^c \\ y_n^c \end{bmatrix} + h \begin{bmatrix} 1/3 & 2/3 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 6f_{n-1}^c \\ 6f_n^c \end{bmatrix} \tag{3.2}$$

and

$$\begin{bmatrix} y_{n+1}^c \\ y_{n+2}^c \end{bmatrix} = \begin{bmatrix} -5/23 & 28/23 \\ -4/23 & 27/23 \end{bmatrix} \begin{bmatrix} y_{n-1}^c \\ y_n^c \end{bmatrix} + h \begin{bmatrix} 11/69 & -2/69 \\ 6/23 & 1/23 \end{bmatrix} \begin{bmatrix} 6f_{n+1}^p \\ 6f_{n+2}^p \end{bmatrix} \tag{3.3}$$

or in matrix formulation

$$Y_m^p = AY_{m-1}^c + hBF_{m-1}^c \tag{3.4}$$

and

$$Y_m^c = A^*Y_{m-1}^c + HB^*(AY_{m-1}^c + HBY_{m-1}^c) = (A^* + HB^*A + H^2B^*B)Y_{m-1}^c \tag{3.5}$$

Define

$$P(H) = A^* + HB^*A + H^2B^*B \tag{3.6}$$

The locus of Formula 3.6; $P(H)$, it determines the boundary of the stability region and is given in Figure 3.3.

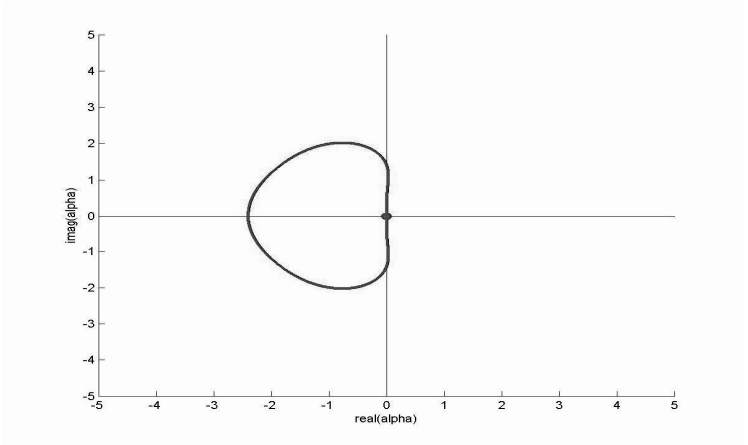


Figure 3.3:Locus plot of a block $PE(CE)^1$ by Formulas 2.2 and 2.3
The region is the interior of the blue line

Define $H^* = \sup\{-H \mid \rho(P(H) < 1, H = \lambda h, \mathrm{Re}(\lambda) < 0\}$. In Figure 3.3, the intercept is about -2.5. According to [4,7,8,9], this intercept is among the largest of the PECE schemes, and especially our method is a block PECE scheme. That makes our scheme has a better control of the stepsize in the integration.

Before ending this section, we like to point out one special observation of our formula 3.3 which has great similarity to EBD method [2]. The first component of formula 3.3 can be written as

$$y_{n+1} - \frac{28}{23} y_n + \frac{5}{23} y_{n-1} = \frac{22}{23} f_n - \frac{4}{23} f_{n+1}, \qquad (3.7)$$

and the same order formula from EBD can be written as

$$y_{n+1} - \frac{28}{23} y_n + \frac{5}{23} y_{n-1} = \frac{22}{23} f_n - \frac{4}{23} f_{n+1},$$

which is exactly the same as Formula 3.7. But, in Formula 3.3, since it is a block formula, so we have an additional formula for further extended step-point y_{n+2}, and it makes our formula to be a block extended step-point backward differentiation formulas. It may increase the competitive in computation.

4. Numerical Experiments

In the numerical schemes, Formulas 2.2 ~2.7 can be implemented in either Newton iteration by the implicit formulas, such as Formulas 2.3, 2.5 and 2.6; or by a predictor-corrector scheme, such as Formulas 2.2 and 2.3; 2.4 and 2.5, or 2.6 and 2.7, etc. We will show few examples to see the effect of applying a variable stepsize selection strategy, and a Block Predictor-Corrector numerical scheme on mildly stiff ODEs. Example 1 is a mildly stiff ODE. The initial stepsize is 10^{-3}. Numerical output

is given in Figure 4.1, and the absolute error is between $10^{-8} \sim 10^{-10}$ and is given in Figure 4.2. Example 2 is a nonlinear mildly stiff ODEs. Numerical output is given in Figure 4.3, and the absolute error is between $10^{-8} \sim 10^{-11}$ and is given in Figure 4.4.

Example 1: A mild stiff ODE. $y' = -\lambda y$, $y(0) = 1$, $t \in [0,5]$.
The exact solution is $y = e^{-\lambda t}$. Case 2: $\lambda = 60$

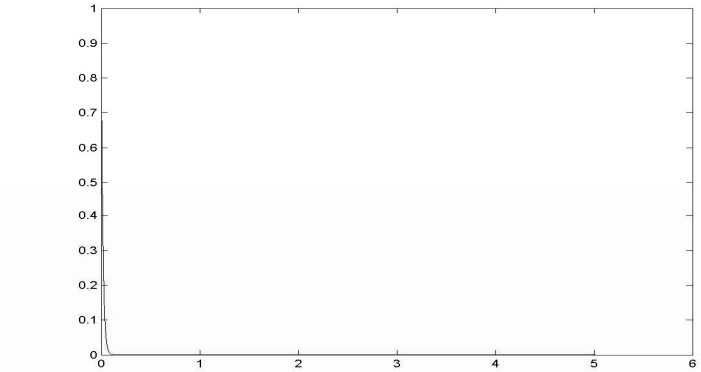


Figure 4.1: Block PECE numerical solution with $\lambda = 60$

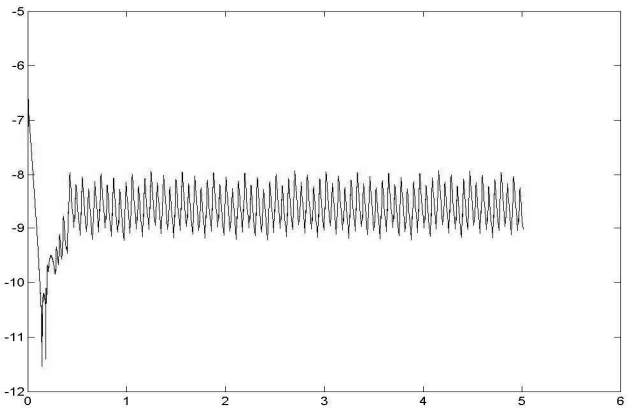


Figure 4.2: Absolute error log plot with variable stepsize when $\lambda = 60$
(the y-axis is by log scale and is about $10^{-5} \sim 10^{-12}$)

Example 2:

$$\begin{aligned} y_1' &= -y_2 - y_1 y_3 / r \\ y_2' &= y_1 - y_2 y_3 / r \\ y_3' &= y_1 / r \end{aligned}$$, where $r = \sqrt{y_1^2 + y_2^2}$, $\begin{aligned} y_1(0) &= 3 \\ y_2(0) &= 0, \quad t \in [0,20] \\ y_3(0) &= 0 \end{aligned}$

True solutions: $y_1 = (2 + \cos(t))\cos(t)$
$y_2 = (2 + \cos(t))\sin(t)$
$y_3 = \sin(t)$

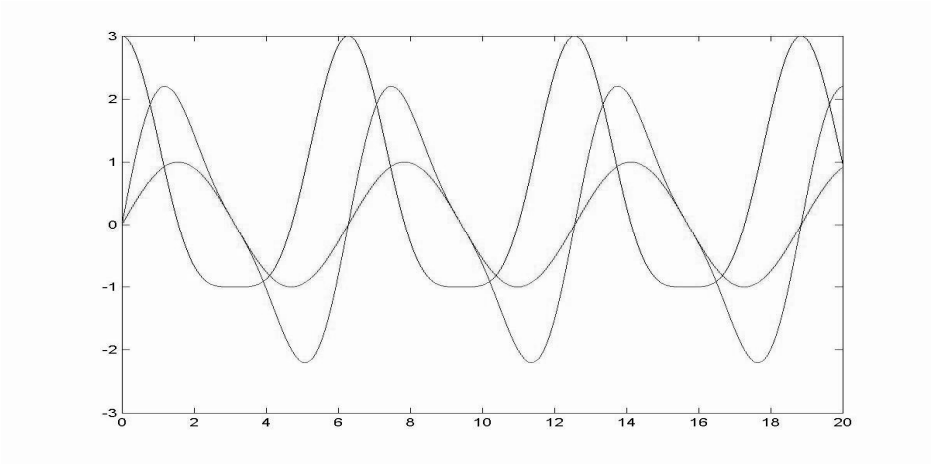


Figure 4.3: Block PECE numerical solutions

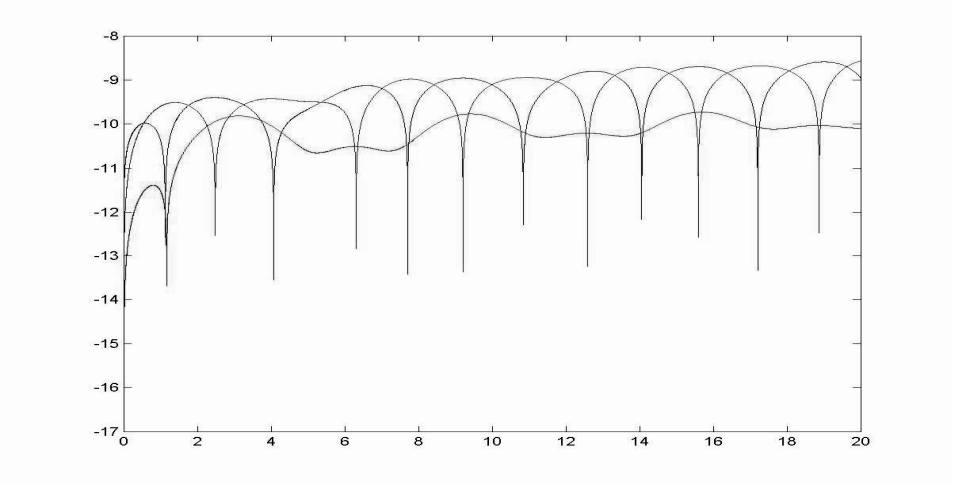


Figure 4.4: Absolute error of Block PECE numerical solutions with fixed stepsize (the y-axis is by log scale and is about $10^{-9} \sim 10^{-12}$)

We also show a stiff ODE example by using implicit formulas, e.g., Formulas 2.3 (or 2.6).

Example 3:

$$\begin{aligned} y_1' &= -0.04y_1 + 10^4 y_2 y_3 & y_1(0) &= 1 \\ y_2' &= 0.04y_1 - 10^4 y_2 y_3 - 3\times 10^7 y_2^2 \quad \text{with} & y_2(0) &= 0 \\ y_3' &= 3\times 10^7 y_2^2 & y_3(0) &= 0 \end{aligned}$$

A fixed stepsize $h = 10^{-3}$ was used to solve this example by applying Formula 2.3 with Newton iteration. A variable stepsize solver ode15s from MATLAB was also implemented, and its numerical solutions are used as comparison. Figures 4.5 - 4.7 contain solutions solved by these two schemes. From the output, solutions by ode15s

from MATLAB and our scheme match well. Since our scheme is currently implemented by fixed stepsize and ode15s is by variable stepsize, the estimation of absolute error will be not given shown.

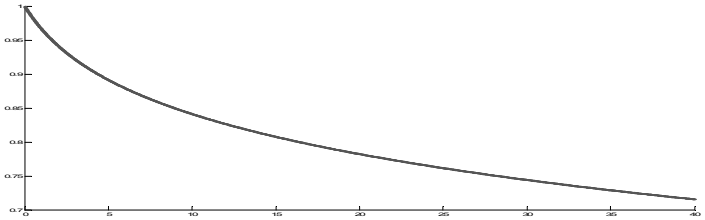


Figure 4.5: Numerical output of $y_1(x)$ obtained by 2 schemes

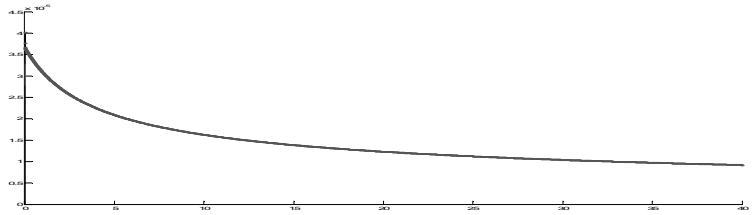


Figure 4.6: Numerical output of $y_2(x)$ obtained by 2 schemes

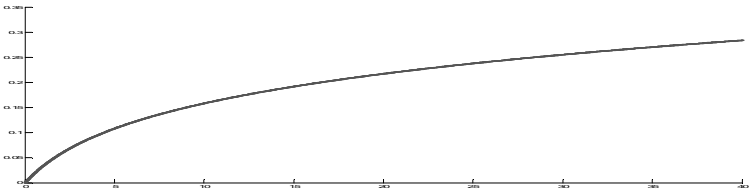


Figure 4.7: Numerical output of $y_3(x)$ obtained by 2 schemes

5. Conclusion

In this paper, we have shown a new Block extended backward differentiation formulas with different stages s and steps m. Some absolute stability regions of these formulas have been sketched, the results show that the region are either A stable or $A(\alpha)$ stable. In addition, stability region of implement a predictor corrector scheme by these block formula is also established, their intercept of stability region is the best among some known literatures. Numerical results by implementing a predictor-corrector schemes with stepsize selection strategy to some mildly stiff ODEs are obtained, the numerical results show that the method is both effective and accurate regarding to obtain r new values at each integration step and its rate of convergence. In addition, an implicit implementation is also given, their numerical results match well with the output of the famous MATLAB numerical method ode15s. Numerical solutions of stiff ODEs and some linear DAEs with constant and time dependent coefficients matrices by implicit block scheme will be studied in the future.

References

1. Bulatov, M.V., “Construction of a One-Stage L-Stable Second Order Method”, Differential Equations, Vol. 39, No. 4, pp. 593-595.
2. Cash J.R., On The Integration of Stiff Systems of ODEs Using Extended Backward Differentiation Formulas, Numer. Math., 34, (1980) pp235-246.
3. Cash J.R., Modified Extended Backward Differentiation Formulae for the Numerical Solution of Stiff Initial Value problems in ODEs and DAEs, Journal of Computational and Applied Mathematics, 125, (2000) pp. 117-130.
4. Fang, Huei-Jen, “On Adaptive Block Predictor-Corrector Methods for ODEs”, Doctoral Dissertation, National Chung Cheng University, Taiwan, 2001.
5. Hairer E. and Wanner G., Solving Ordinary Differential Equations I Stiff and Differential Algebraic Problems, Springer Berlin, 1996.
6. Gear, C.W., Algorithm 407, Difsub for Solution of Ordinary Differential Equations, Comm. ACM 14,(1971) pp.185-190.
7. Lee, M.G and Song, R.W., A New Block Method for Stiff Differential equations, 2009 International Conference on Scientific Computation and Differential equations (SciCADE 2009), Beijing, China.
8. Majid, Zanariah Abdul and Suleiman, Mohamed Bin, Implementation of Four-Point Fully Implicit Block Method for Solving Ordinary Differential equations, Applied Mathematics and Computation 184, (2007), pp 514-522.
9. Shampine, L. F. and Watts, H. A., “Block Implicit One-Step Methods”, Math. Comp. 23, (1969), pp. 731-740.

PGB: A package for computing parametric polynomial systems

Katsusuke Nabeshima

Graduate School of Information Science and Technology, Osaka University,
Machikaneyama 1-1, Toyonaka 560-0043, Osaka, JAPAN.
Japan Science and Technology Agency, CREST,
Sanbancho, Chiyoda-ku, Tokyo, 102-0075, JAPAN.
nabeshima@math.sci.osaka-u.ac.jp

Abstract

We describe a new software package, named PGB, for computing parametric polynomial systems and related objects. PGB is applicable to a wide range of expressions for many of which there has not been any software available up to now. The purpose of this paper is to illustrate how to solve using that package.

1 Introduction

In this paper, we shall introduce a new Risa/Asir[17] package for computing parametric polynomial systems. The main part of this package is based on the theory of parametric Gröbner bases. (That why, this package is called PGB (**P**arametric **G**röbner **B**ases).) The use of Gröbner basis computations for treating systems of polynomial equations has become an important tool in many areas, reaching from pure mathematics to industrial applications. For every concept and construction in computer algebra the question of uniformity in the input parameters is crucial both from a theoretical and a practical viewpoint. This applies in particular to the concept of Gröbner bases. Recently, Gröbner bases for parametric polynomials have been actively investigated. For example, one can see several papers for Gröbner bases of a polynomial ideal with parametric coefficients in [1, 3, 7, 12, 13, 14, 19, 20, 22, 23]. The goal of this article is to describe *what* PGB can do and to explain how to get it do something, but we do not comments on *how* it obtains its results. The paper is intended as a guide for potential user of the package. The underlying algorithms are described elsewhere [13, 15, 19].

The outline of the paper is as follows: Section 2 presents the basic notations for polynomial rings over a filed, rings of differential operators, and modules. Section 3 describes how to solve problems of parametric polynomial systems. Section 4 describes how to solve a system of parametric linear equations. In the section 5, we conclude this paper.

PGB has been implemented in the computer algebra system Risa/Asir [17] (`http://www.math.kobe-u.ac.jp/Asir/asir.html`). It is available free of charge for any non-commercial user and can be obtained from

`http://www.math.sci.osaka-u.ac.jp/~nabeshima/PGB/`

or upon request from the author. In the web-page, one can download the manual of PGB, too. **This paper is not the manual**, the purpose of this paper is to illustrate how to solve using that package.

*Correspondence to: Graduate School of Information Science and Technology, Osaka University, Machikaneyama 1-1, Toyonaka 560-0043, Osaka, JAPAN. Tel:+81-6-6850-5551

2 Notations

Throughout this paper, we assume that K and L are fields such that L is an extension of K. $\mathbb{Q}$ and $\mathbb{C}$ denote as the set the field of rational numbers and the field of complex numbers, respectively. The notations $\bar{X}$ and $\bar{A}$ will be used as an abbreviation for the set of n variables $\{x_1, \ldots, x_n\}$ and m variables $\{a_1, \ldots, a_m\}$, respectively. I.e., $K[\bar{X}] := K[x_1, \ldots, x_n]$, $K[\bar{A}] := K[a_1, \ldots, a_m]$. Let $f_1, \ldots, f_s$ be polynomials in $K[\bar{A}]$. Then we set $\mathbb{V}(f_1, \ldots, f_s) = \{(b_1, \ldots, b_m) \in L^m | f_i(b_1, \ldots, b_m) = 0 \text{ for all } 1 \leq i \leq s\}$, and $\mathbb{V}(0) := L^m$, $\mathbb{V}(1) := \emptyset$. Let I be an ideal in $K[\bar{X}]$ and $\succ$ a term order. Then, $\mathrm{lm}(I) := \{\mathrm{lm}(g) | g \in I\}$ where $\mathrm{lm}(g)$ is the leading monomial of g with respect to $\succ$.

For arbitrary $\bar{a} \in L^m$, we can define the canonical **specialization homomorphism** $\sigma_{\bar{a}} : K[\bar{A}] \to L$ induce by $\bar{a}$, and we can naturally extend it to $\sigma_{\bar{a}} : K[\bar{A}, \bar{X}] \to L[\bar{X}]$.

◯ **Rings of differential operators**
Let $\partial_i = \frac{\partial}{\partial x_i} : K[\bar{X}] \to K[\bar{X}]$ be the partial derivative by x_i, $1 \leq i \leq n$. Let $K[\bar{X}, \bar{D}] := K[x_1, \ldots, x_n, \partial_1, \ldots, \partial_n]$ be the rings of differential operators (and variables) with coefficients in K. $K[\bar{X}, \bar{D}]$ has the commutation rules

$$x_i x_j = x_j x_i,\ \partial_i \partial_j = \partial_j \partial_i,\ \partial_i x_j = x_j \partial_i, \quad \text{for } i \neq j, \text{ and } \partial_i x_i = x_i \partial_i + 1.$$

It is well-known that $K[\bar{X}, \bar{D}]$ is a left-Noetherian associative K-algebra. By an "ideal in $K[\bar{X}, \bar{D}]$" we always mean a left-ideal of $K[\bar{X}, \bar{D}]$.

◯ **Modules**
Let $e_1, \ldots, e_r$ be the canonical basis of the free module $K[\bar{X}]^r := \bigoplus_{i=1}^r K[\bar{X}] e_i$. I.e, for each $i = 1, \ldots, r$,

$$e_i = \big(0, \ldots, 0, \overset{i\text{th}}{1}, 0, \ldots, 0\big)^T \in K[\bar{X}]^r$$

denotes the i-th canonical basis vector of $K[\bar{X}]^r$ with 1 at the i-th place. (Let A be a matrix. In this paper, the transposed matrix of A is written as A^T.) Two module orders are of particular practical interest: POT (position over term) and TOP (term over position) which follows [4].

Let Z be a $K[\bar{X}]$, $K[\bar{X}, \bar{D}]$ or $K[\bar{X}]^r$ and $f_1, \ldots, f_s \in Z$. Then we set $\langle f_1, \ldots, f_s \rangle = \{\sum_{i=1}^s h_i f_i \,|\, h_1, \ldots, h_s \in Z\}$. The crucial fact is that $\langle f_1, \ldots, f_s \rangle$ is an ideal (or a module) in Z. We call $\langle f_1, \ldots, f_s \rangle$ the ideal (or module) generated by $f_1, \ldots, f_s$.

3 Parametric Gröbner Bases in various domains

PGB has commands for computing parametric Gröbner bases in commutative polynomial rings over fields, rings of differential operators and modules. These computational commands are the main achievements of this package. In order to apply these commands, we can solve some parametric problems, like normal Gröbner bases.

In general, parametric Gröbner bases (with parametric coefficients) have two types. One is called "comprehensive Gröbner bases". The other is called "comprehensive Gröbner systems". Comprehensive Gröbner bases and systems for parametric polynomial ideals were introduced, constructed, and studied by Weispfenning in 1992 [20]. Since then comprehensive Gröbner bases and systems have been studied and implemented in the several computer

algebra systems. After Weispfenning's paper was published, Dolzman and Sturm have implemented and published a software [2]. However, there was no big development about comprehensive Gröbner bases and systems for ten years. Recently, the big developments were made by Montes, Sato, Suzuki, Weispfenning and the author, as follows;

- Montes published the new algorithm for computing comprehensive Gröbner systems and its software in 2002 and 2006 [12, 13]. .
- Suzuki and Sato presented an alternative definition of comprehensive Gröbner bases in terms of Gröbner bases in polynomial rings over commutative von Neumann regular rings in 2003 [18]. This Gröbner basis is called "alternative comprehensive Gröbner basis (ACGB)". Alternative comprehensive Gröbner bases have the following nice properties, which do not hold in standard comprehensive Gröbner bases;
  - There is a canonical form of an alternative comprehensive Gröbner basis in a natural way.
  - We can use reductions of an alternative comprehensive Gröbner basis.
- Weispfenning presented a concept of canonical comprehensive Gröbner bases under very general assumptions on the parameter ring in 2002 and 2003 [21]. After this paper was published, this result was applied for improving Montes' algorithm by Montes [12].
- Suzuki and Sato published the new algorithms for computing comprehensive Gröbner bases and systems in 2006 [19]
- The author has improved the Suzki-Sato algorithm (not ACGB) for computing compressive Gröbner systems in 2007 [15].

Now, there exist the software packages or programs for computing comprehensive Gröbner bases and systems; REDLOG in Reduce, Montes' program in Maple, Suzuki's program in Maple, Risa/Asir, Mathematica, singular and the author's this PGB in Risa/Asir.

In the package PGB, the Suzuki-Sato algorithm[19] and Nabeshima's techniques[15] are applied for computing parametric Gröbner bases, because the Suzuki-Sato algorithms and the Nabeshima's techniques are faster than other existing algorithms. Moreover, in the structure point of view, these algorithms are simpler than other existing algorithms. This means that one can easily implement them. That's why we applied the algorithms.

Before describing the package, first we introduce the definitions of "comprehensive Gröbner bases" and "comprehensive Gröbner systems".

Let Z be a $K[\bar{A}][\bar{X}]$, $K[\bar{A}][\bar{X},\bar{D}]$ or $(K[\bar{A}][\bar{X}])^r$. Fix a term or module order.

Definition 3.1 (Comprehensive Gröbner bases). *Let F and G be subsets in Z. $G \subset \langle F\rangle$ is called a* **comprehensive Gröbner basis** *for $\langle F\rangle$ if $\forall \bar{a} \in L^m$, $\sigma_{\bar{a}}(G)$ is a Gröbner basis for $\langle \sigma_{\bar{a}}(F)\rangle$ in $\sigma(Z)$.*

Definition 3.2 (Comprehensive Gröbner systems). *Let F be a subset of Z, $\mathcal{A}_1,\ldots,\mathcal{A}_l$ algebraically constructible subsets of L^m and $G_1,\ldots,G_l$ subsets of Z. A finite set $\mathcal{G} = \{(\mathcal{A}_1,G_1),\ldots,(\mathcal{A}_l,G_l)\}$ of pairs is called a* **comprehensive Gröbner system** *for $\langle F\rangle$ if $\sigma_{\bar{a}}(G_i)$ is a Gröbner basis of the ideal $\langle \sigma_{\bar{a}}(F)\rangle$ in $\sigma(Z)$ for each $i = 1,\ldots,l$ and $\forall \bar{a} \in \mathcal{A}_i$. Each $(\mathcal{A}_i,G_i)$ is called a* **segment** *of $\mathcal{G}$.*

Example 3.3. *Let* $F = \{ax^2y+y, bx^2y^2+ax+y\} \subset \mathbb{Q}[x,y]$, a, b *parameters,* x, y *variables and* $\succ$ *the purely lexicographic order such that* $x \succ y$. *Then, a comprehensive Gröbner basis for* $\langle F\rangle$ *with respect to* $\succ$ *is*

$$G = \Big\{ bxy^3 - axy^2 + ay, a^2x - by^2 + ay, -b^2y^5 + 2aby^4 - a^2y^3 - a^3y, -bx^3y^3 - xy^2 + y,$$
$$bx^2y^2 + ax + y, bx^2y^4 + bxy^3 + y^3 + ay, ax^2y + y \Big\}.$$

Even if we substitute arbitrary values for the parameters a *and* b *of the set* G, *the set* $\sigma(G)$ *is always a Gröbner basis for* $\langle\sigma(F)\rangle$ *with respect to* $\succ$.

A comprehensive Gröbner system for $\langle F\rangle$ *with respect to* $\succ$ *is*

$$\mathcal{G}_1 = \big\{ \big(\mathbb{Q}^2 \setminus \mathbb{V}(a,b), \{a^2x - by^2 + ay, -b^2y^5 + 2bay^4 - a^2y^3 - a^3y\}\big), \big(\mathbb{V}(a,b), \{y\}\big) \big\}.$$

The union of the all parameters' spaces is $\mathbb{Q}^2$. *i.e.,* $\mathbb{Q}^2 = \Big(\mathbb{Q}^2 \setminus \mathbb{V}(a,b)\Big) \cup \big(\mathbb{V}(a,b)\Big)$.

3.1 Commutative polynomial rings

PGB has commands for computing parametric Gröbner bases in commutative polynomial rings. These commands are based on the Suzuki-Sato algorithms and Nabeshima's computational techniques. We do not aim in this article at explaining in detail the algorithm that PGB is bases on. We illustrate how to solve using PGB.

◯ **Parametric Gröbner bases**

There exist several commands for computing comprehensive Gröbner bases and systems in PGB. Each command has different techniques for getting optimal outputs. One have to select one command to compute parametric Gröbner bases.

Let $I = \langle ax^2y^2+bxy+2, bx+ay+2\rangle$ be an ideal in $\mathbb{C}[x,y]$ with parameters a, b, and $\succ$ be the purely lexicographic order such that $x \succ y$. We select a command "`cgs1`" for obtaining a comprehensive Gröbner system. Then, `cgs1([a*x^2*y^2+b*x*y+2,b*x+a*y+2],[a,b],[x,y],1,2)` outputs the following as a comprehensive Gröbner system for I with respect to $\succ$.

```
[394] cgs1([a*x^2*y^2+b*x*y+2,b*x+a*y+2],[a,b],[x,y],1,2);
[b]==0,  [a]!=0,
[-2*x^2-a,a*y+2]
[b,a]==0,  [1]!=0,
[1]
[a]==0,  [b]!=0,
[b*x+2,-y+1]
[0]==0,  [b*a]!=0,
[-a^3*y^4-4*a^2*y^3+(b^2-4)*a*y^2+2*b^2*y-2*b^2,b*x+a*y+2]
Number of segments is 4
```

This output means

$$\begin{cases} \{ay+2, -2x^2-a\}, & \text{if } \mathbb{V}(b)\setminus\mathbb{V}(a), \\ \{1\}, & \text{if } \mathbb{V}(a,b), \\ \{-y+1, bx+2\}, & \text{if } \mathbb{V}(a)\setminus\mathbb{V}(b), \\ \{-a^3y^4 - 4a^2y^3 + (b^2-4)ay^2 + 2b^2y - 2b^2, bx+ay+2\} & \text{if } \mathbb{C}^2\setminus\mathbb{V}(ab). \end{cases}$$

Next, let see that PGB computes comprehensive Gröbner bases. We choose a command "cgb1" for getting a comprehensive Gröbner bases. Then, `cgb1([a*x^2*y^2+b*x*y+2,b*x+a*y+2],[a,b],[x,y],2)` outputs the following as a comprehensive Gröbner basis for I with respect to $\succ$.

```
[394] cgb1([a*x^2*y^2+b*x*y+2,b*x+a*y+2],[a,b],[x,y],2);
[a*y^2*x^2-a*y^2-2*y+2,(a^2*y^3+2*a*y^2)*x+b*a*y^2+2*b*y-2*b,a^3*y^4+4*a^2*
y^3+(-b^2+4)*a*y^2-2*b^2*y+2*b^2,b*x+a*y+2,-a*y^2*x^2+a*y^2+2*y-2,-b*x-a*y-
2,(-b*a*y+2*b)*x^3+4*x^2+b*a*y*x+2*a]
```

If we substitute any values for the parameters a, b of the output above, then the set computed always a Gröbner basis for $\sigma(I)$ with respect to $\succ$ where σ is the specialization defined the substitution. However, note that this Gröbner basis is not always the reduced Gröbner basis for $\sigma(I)$ with respect to $\succ$.

◯ Applications

There are a lot of applications of parametric Gröbner bases, like normal Gröbner bases. One of the applications is a classification of dimensions for parametric ideals. Dimensions of a parametric ideal are dependent on the values of parameters. For example, let $I = \langle ax, by^2 + y\rangle$ be an ideal in $\mathbb{C}[x, y]$ with parameter a and b. In this case, if $a = 0, b \neq 0$, then the dimension of this ideal is clearly 1. However, if $ab \neq 0$, then the dimension of this ideal is clearly 0. By the values of the parameters, the dimension of this parametric ideal is changed. Now, we have a question. **How can we classify dimensions of a parametric ideal by parameters' spaces?** The theory of parametric Gröbner bases is able to solve this question, and PGB has a command to solve this question.

The Suzuki-Sato algorithms and Nabeshima's computational techniques are based on the theory of stability of ideals [5, 6]. This means that the algorithms find a parameters' space which lets the ideal stable. Here, let $\mathcal{G} = \{(\mathcal{A}_1, G_1), \ldots, (\mathcal{A}_l, G_l)\}$ be an output of the Suzuki-Sato algorithm (or Nabeshima's) for computing comprehensive Gröbner systems where $\mathcal{A}_1, \ldots, \mathcal{A}_l$ are parameters' spaces and $G_1, \ldots, G_l$ are set of parametric polynomials. Then, $\forall \alpha \in \mathcal{A}_i$, $\sigma_\alpha(\mathrm{lm}(G_i))$ does not have any zero elements, for each $i = 1, \ldots, l$. That is, all leading coefficients of the polynomials can not become zero under the specialization [15, 19]. This fact leads us to compute a classification of dimensions for a parametric ideal I, because it is same as a classification of $\mathrm{lm}(I)$. This computational algorithm is the following;

1. Compute a comprehensive Gröbner system $\mathcal{G}$.
2. Compute the dimension of each segment of the comprehensive Gröbner system $\mathcal{G}$.

PGB has a command "class_dim_para" for a classification of dimensions for a parametric ideal. Let $I = \langle ax, by^2 + y\rangle$ in $\mathbb{C}[x, y]$ with parameter a and b. The command works as follows;

```
[395] class_dim_para([a*x,b*y^2+y],[a,b],[x,y],0);
[[0],[b*a]]                              [[a],[a,b]]
the dim. is  0                           the dim. is  1
[[b],[a,b]]                              [[a,b],[1]]
the dim. is  0                           the dim. is  1
```

This output means;
if the parameters take the values from $\mathbb{C}^2 \setminus \mathbb{V}(ba)$, then the dimension of the ideal I is 0,
if the parameters take the values from $\mathbb{V}(b) \setminus \mathbb{V}(a, b)$, then the dimension of the ideal I is 0,

if the parameters take the values from $\mathbb{V}(a)\setminus\mathbb{V}(a,b)$, then the dimension of the ideal I is 1, if the parameters take the values from $\mathbb{V}(a,b)$, then the dimension of the ideal I is 1.

3.2 Rings of differential operators

Here we describe how to get parametric Gröbner bases in $K[\bar{X},\bar{D}]$ a ring of differential operators by PGB. That is, PGB has commands for computing non-commutative parametric Gröbner bases. In [9, 10], Kredel and Weispfenning studied parametric Gröbner bases for non-commutative polynomials. In these papers, they applied the Weispfenning method [20] to compute parametric Gröbner bases. However, they have not implemented them. **Now, PGB is the only one existing program for computing parametric Gröbner bases in rings of differential operators.** The commands for computing parametric Gröbner bases in rings of differential operators, are bases on the Suzuki-Sato algorithms. The author has generalized the algorithms to the case rings of differential operators, and implemented them in Risa/Asir.

◯ **Parametric Gröbner bases**
As well as the case of commutative polynomial rings, there exist several commands for computing comprehensive Gröbner bases and systems in PGB. Each command has different techniques for getting optimal outputs. One have to select one command to compute parametric Gröbner bases in Weyl algebra.

Let $I=\langle ax_1\partial_1^2\partial_2+(a+1)x_1x_2\partial_2, x_2^2\partial_2+bx_1, \partial_1\partial_2^2\rangle$ be an ideal in $\mathbb{C}[x_1,x_2,\partial_1,\partial_2]$ with parameters a,b, and $\succ$ be the purely lexicographic order such that $x_1\succ x_2\succ\partial_1\succ\partial_2$. (We have the rules $x_ix_j=x_jx_i,\ \partial_i\partial_j=\partial_j\partial_i,\ \partial_ix_j=x_j\partial_i,\ $ for $i\neq j,$ and $\partial_ix_i=x_i\partial_i+1$. for $i=1,2$.) We select a command "`cgs1`" for obtaining a comprehensive Gröbner system.

```
[396] cgsw([a*x1*d1^2*d2+(a+1)*x1*x2*d2,x2^2*d2+b*x1,d1*d2^2],[a,b],[[x1,
x2],[d1,d2]],1,2);
[a+1]==0,  [b]!=0,
[x1,d2]
[b,a+1]==0,  [1]!=0,
[x2^2*d2,d1*d2]
[b]==0,  [a+1]!=0,
[x2*d2,d1*d2]
[0]==0,  [b*a+b]!=0,
[x1,d2]
Number of segments is 4
```

This output means the following

$$\begin{cases} \{x_1,\partial_2\}, & \text{if } \mathbb{V}(a+1)\setminus\mathbb{V}(b),\\ \{x_2^2\partial_2,\partial_1\partial_2\}, & \text{if } \mathbb{V}(b,a+1),\\ \{x_2\partial_2,\partial_1\partial_2\} & \text{if } \mathbb{V}(b)\setminus\mathbb{V}(a+1),\\ \{x_1,\partial_2\} & \text{if } \mathbb{C}^2\setminus\mathbb{V}(ba+b). \end{cases}$$

Next, we choose a command "`cgbw`" for getting a comprehensive Gröbner basis for I in the ring of differential operators. Then, `cgbw` outputs the following as a comprehensive Gröbner basis for I with respect to $\succ$.

```
[397] cgbw([a*x1*d1^2*d2+(a+1)*x1*x2*d2,x2^2*d2+b*x1^2,d1*d2^2],[a,b],[[x1,
x2], [d1,d2]],2);
[x2^2*d2+b*x1^2,x2^2*d2^2+2*x2*d2,(a+1)*x2*d2,d1*d2,b*d2]
```

◯ **Applications**
Here, we give one simple example for applying the elimination property of Gröbner bases to systems of parametric linear differential equations.

Let $\{2xy''+axy'=0, xy'''+bx^2y'-bxy=0, xy''-axy=0\}$ be a set of linear differential equations with parameters a, b. This system of linear ordinary differential equations can be written as $\{(2x\partial^2+ax\partial)y=0, (x\partial^3+bx^2\partial-bx)y=0, (x\partial^2-ax)y=0\}$ where ∂ is the partial derivative $\frac{\partial}{\partial x}$. Set $f_1=2x\partial^2+ax\partial$, $f_2=x\partial^3+bx^2\partial-bx$, $f3=x\partial^2-ax$ in $\mathbb{C}[x,\partial]$. Now, we can compute a comprehensive Gröbner system for $\langle f_1, f_2, f_3\rangle$ in $C[x,\partial]$ by using PGB. Then, PGB outputs the following as the comprehensive Gröbner system.

```
[396] cgsw([2*x*d^2+a*x*d,x*d^3+b*x^2*d-b*x,x*d^2-a*x],[a,b],[[x],[d]],1,2);
[b,a]==0,  [1]!=0,
[d^2]
[a]==0,  [b]!=0,
[b*x*d-b,d^2]
[0]==0,  [a]!=0,
[1]
Number of segments is 3
```

This output means the following;

$$\begin{cases} \{\partial^2\}, & \text{if } \mathbb{V}(a,b), \\ \{bx\partial-b, \partial^2\}, & \text{if } \mathbb{V}(a)\setminus\mathbb{V}(b), \\ \{1\}, & \text{if } \mathbb{C}^2\setminus\mathbb{V}(a). \end{cases}$$

Hence, the system of linear differential equations can be reduced to :

$$\begin{cases} \{y''\}, & \text{if } \mathbb{V}(a,b), \\ \{bxy'-b, y''\}, & \text{if } \mathbb{V}(a)\setminus\mathbb{V}(b), \\ \{y\}, & \text{if } \mathbb{C}^2\setminus\mathbb{V}(a). \end{cases}$$

Therefore;

(1) In case the values of parameters a, b are from $\mathbb{V}(a,b)$, then $y''=0$. Hence, $y=c_1x+d_1$ is the general solution of the system where $c_1, d_1 \in \mathbb{C}$.

(2) In case the values of parameters a, b are from $\mathbb{V}(a)\setminus\mathbb{V}(b)$, then $y''=0$. Hence, $y=c_2x+d_2$ is the general solution of the system where $c_2, d_2 \in \mathbb{C}$.

(3) In case the values of parameters a, b are from $\mathbb{C}^2\setminus\mathbb{V}(a)$, then $y=0$. That' is, we have only the trivial solution $y=0$.

3.3 Modules

In this subsection, we describe the command for computing comprehensive Gröbner bases and systems **for modules**. In several papers and books [4, 8, 11], an algorithm for computing Gröbner bases for $K[\bar{X}]$-modules and its properties were introduced. Theoretically, Gröbner basis algorithms admit natural extensions to modules. However, especially in the parametric situation, complexity is an important issue. An efficient algorithm for computation of comprehensive Gröbner bases over polynomial rings, was proposed by Suzuki and Sato[19]. The author generalized the Suzuki-Sato algorithms to the module [16] and implemented them in Risa/Asir. Now, PGB is the only one existing program to compute parametric Gröbner bases for modules.

◯ **Parametric Gröbner bases**
As well as the case of commutative polynomial rings, there exist several commands for computing comprehensive Gröbner bases and systems in PGB. Each command has different techniques for getting optimal outputs. One have to select one command to compute parametric Gröbner bases for modules.

Let x, y be variables, a, b parameters and $\succ_{lex}$ the purely lexicographic order such that $x \succ_{lex} y$. We consider $f_1 = \begin{pmatrix} ax - bx + 1 \\ ax^2y + ax + b \end{pmatrix}$ and $f_2 = \begin{pmatrix} by + a \\ bx^2 + bx + 2 \end{pmatrix}$ in $\mathbb{Q}[x, y]^2$ with parameters a, b. Then, our the command "`cgs_m`" outputs the following list which is a comprehensive Gröbner system for $\langle f_1, f_2 \rangle$ with respect to POT with $\succ_{lex}$.

```
[397] cgs_m([[a*x-b*x+1,a*x^2*y+a*x+b],[b*y+a,b*x^2+b*x+2]],[a,b],[x,y],1,p,
2);
[a-b]==0,  (b)!=0,
[0,(b^2*y^2+b^2*y-b)*x^2+(b^2*y+b^2-b)*x+b^2*y+b^2-2]
[1,b*y*x^2+b*x+b]

[b,a]==0,  (1)!=0,
[0,1]
[1,0]

[b]==0,  (a)!=0,
[0,-a^2*y*x^2+(-a^2+2*a)*x+2]
[1,a*y*x^2+(a-2)*x]

[0]==0,  (b)*(a-b)!=0,
[0,(-b*a+b^2)*x^3+(b*a*y^2+a^2*y-b*a+b^2-b)*x^2+(b*a*y+a^2-2*a+b)
*x+b^2*y+b*a-2]
[b*y+a,b*x^2+b*x+2]
[(a-b)*x+1,a*y*x^2+a*x+b]
```

The output means the following:
If $\mathbb{V}(a-b) \setminus \mathbb{V}(b)$, then $\left\{ \begin{pmatrix} 0 \\ (*1) \end{pmatrix}, \quad \begin{pmatrix} 1 \\ byx^2 + bx + b \end{pmatrix} \right\}$.
If $\mathbb{V}(a, b)$, then $\left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$.
If $\mathbb{V}(b) \setminus \mathbb{V}(a)$, then $\left\{ \begin{pmatrix} 0 \\ -a^2yx^2 + (-a^2 + 2a)x + 2 \end{pmatrix}, \begin{pmatrix} 1 \\ ayx^2 + (a-2)x \end{pmatrix} \right\}$.
If $\mathbb{C}^2 \setminus \mathbb{V}((a-b)b)$, then $\left\{ \begin{pmatrix} 0 \\ (*2) \end{pmatrix}, \begin{pmatrix} by + a \\ bx^2 + bx + 2 \end{pmatrix}, \begin{pmatrix} (a-b)x + 1 \\ ayx^2 + ax + b \end{pmatrix} \right\}$,
where $(*1) = (b^2y^2 + b^2y - b)x^2 + (b^2y + b^2 - b)x + b^2y + b^2 - 2$ and $(*2) = (-ba + b^2)x^3 + (bay^2 + a^2y - ba + b^2 - b)x^2 + (bay + a^2 - 2a + b)x + b^2y + ba - 2$.

Next, we see a command "`cgb_m`" for getting a comprehensive Gröbner basis for I. Then, `cgb_m` outputs the following as a comprehensive Gröbner basis for I with respect to the module order.

```
[398] cgb_m([[a*x-b*x+1,a*x^2*y+a*x+b],[b*y+a,b*x^2+b*x+2]],[a,b],[x,y],1,p,
2);
[0,(-b*a+b^2)*x^3+(b*a*y^2+a^2*y-b*a+b^2-b)*x^2+(b*a*y+a^2-2*a+b)*x+b^2*y+
b*a-2]
[b*y+a,b*x^2+b*x+2]
[(a-b)*x+1,a*y*x^2+a*x+b]
[(-b*y-b)*x+1,-b*x^3+(a*y-b)*x^2+(a-2)*x+b]
```

This means;
$\left\{ \begin{pmatrix} 0 \\ (*1) \end{pmatrix}, \begin{pmatrix} by+a \\ bx^2+bx+2 \end{pmatrix}, \begin{pmatrix} (a-b)x+1 \\ ayx^2+ax+b \end{pmatrix}, \begin{pmatrix} (-by-b)x+1 \\ (*2) \end{pmatrix} \right\}$,
where $(*1) := (-ba+b^2)x^3+(bay^2+a^2y-ba+b^2-b)x^2+(bay+a^2-2a+b)x+b^2y+ba-2$ and $(*2) = -bx^3+(ay-b)x^2+(a-2)x+b$.

In general, as parametric Gröbner bases are huge in $K[\bar{X}]^1$, parametric Gröbner bases in $K[\bar{X}]^r$ are huge, too. This means that we need high speed machines and a lot of memory (RAM) in the machines. However, PGB in the both the computer algebra system Risa/Asir still works for a lot of (easy) examples in $K[\bar{X}]^r$ where $r \leq 3$, $|\bar{X}| \leq 3$ and $|\bar{A}| \leq 3$ (OS: WindowsXP, CPU: Pentium M 1.73GHz, Memory: 512MB RAM).

◯ Applications

By studying parametric Gröbner bases for modules, we can solve a lot of parametric problems. For example, consider the problem of syzygy computations. In the ordinary setting, computing a Gröbner basis over a module is closely related to the computation of syzygies [4]. In parametric setting, by computing a comprehensive Gröbner system we can obtain parametric syzygies. PGB has the command "`p_syzygy`" for computing parametric syzygies. By this command, we can solve problems of parametric syzygies as follows.

Let $f_1 = x^2+ay, f_2 = x+b, f_3 = bx+y$ be polynomials in $\mathbb{C}[x,y]$ with parameters a, b. We consider the purely lexicographic order $\succ$ such that $x \succ y$. Then, the command outputs the following as bases of parametric syzygies of (f_1, f_2, f_3) (w.r.t. $\succ$).

```
[399] p_syzygy([[x^2+a*y],[x+b],[b*x+y]],[a,b],[x,y],1,2);
[0]==0,  (b)*(a+1)!=0,
[-x-b,x^2+a*y,0]
[-y+b^2,y*x-b*a*y,-b*x+a*y]
[0,b*x+y,-x-b]

[b]==0,  (1)!=0,
[1,-x,-a]
[0,y,-x]

[a+1]==0,  (b)!=0,
[1,-x,1]
[0,b*x+y,-x-b]
```

This output means the following

$$\begin{cases} \left(\begin{pmatrix} -x-b \\ x^2+ay \\ 0 \end{pmatrix}, \begin{pmatrix} -y+b^2 \\ yx-bay \\ -bx+ay \end{pmatrix}, \begin{pmatrix} 0 \\ bx+y \\ -x-b \end{pmatrix} \right), & \text{if } \mathbb{C}^2 \setminus \mathbb{V}(b(a+1)), \\ \left(\begin{pmatrix} 1 \\ -x \\ -a \end{pmatrix}, \begin{pmatrix} 0 \\ y \\ -x \end{pmatrix} \right), & \text{if } \mathbb{V}(b), \\ \left(\begin{pmatrix} 1 \\ -x \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ bx+y \\ -x-b \end{pmatrix} \right), & \text{if } \mathbb{V}(a+1) \setminus \mathbb{V}(b). \end{cases}$$

4 Solve -systems of parametric linear equations-

PGB has the following command for solving a system of parametric linear equations.

```
solve_para([eqns],[para],[vars])
```

The linear system defined by eqns is solved for the unknowns vars with parameters para. If a solution exists, the solution is returned as a list of equations. If the system is underdetermined, the solver will parameterize the solutions in terms of one or more of the unknowns with parameters. If a solution does not exist in some parameters' spaces, then the solver returns the parameters' spaces.

The author has generalized an ordinary Gaussian elimination to a system of **parametric** linear equations, and implemented it in Risa/Asir. One can solve systems of parametric linear equations as follows.

```
[400] solve_para([a*x+2*y-z-2,2*x-3*y+5*z+3,x+b*y+2*z+2],[a,b],[x,y,z]);
No solution
[[[(5*b+6)*a+2*b+1],[-4*a-3,-b-2]]]

solutions
[[0],[(-25*b^2-60*b-36)*a-10*b^2-17*b-6]]
[[y,(-4*a-3)/((5*b+6)*a+2*b+1)],[z,((-3*b-6)*a-4*b-8)/((5*b+6)*a+2*b+1)],
[x,(7*b+14)/((5*b+6)*a+2*b+1)]]

[[-4*a-3,-b-2],[1]]
[[y,1/4*x+1],[z,-1/4*x]]

[[-5*b-6],[1]]
[[y,20/7*a+15/7],[z,-10/7*b*a-15/14*b+1],[x,-4]]
```

This output means the following.
If the parameters take the values from $\mathbb{V}((5b+6)a+2b+1)\setminus\mathbb{V}(-4a-3,-b-2)$, then the system has no solution.
If the parameters take the values from $\mathbb{C}^2\setminus\mathbb{V}((-25b^2-60b-36)a-10b^2-17b-6)$, then the solution is

$$\left\{y=\frac{-4a-3}{(5b+6)a+2b+1}, z=\frac{(-3b-6)a-4b-8}{(5b+6)a+2b+1}, x=\frac{7b+14}{(5b+6)a+2b+1}\right\}.$$

If the parameters take the values from $\mathbb{V}(-4a-3,-b-2)$, then the solution is

$$\left\{y=\frac{1}{4}x+1, z=-\frac{1}{4}x\right\}$$

which was parameterized.
If the parameters take the values from $\mathbb{V}(-5b-6)$, then the solution is

$$\left\{y=\frac{20}{7}a+\frac{15}{7}, z=-\frac{10}{7}ba-\frac{15}{14}b+1, x=-4\right\}.$$

5 Conclusion

PGB is a software package for dealing with parametric expressions. It is based on the theory of parametric Gröbner bases. One can solve problems of parametric polynomial systems by PGB. Right-now, PGB is the only one existing software for computing parametric Gröbner bases in rings of differential operators, modules and solving systems of parametric linear equations. In this point, this software package is valuable.

References and Notes

[1] Chen, X.F., Li, P., and Wang, D.K. Proving Geometric Theorem by Partitioned-Parametric Gröbner Bases. *ADG 2004, LNAI 3763*, pages 34-43, Springer, 2004.

[2] Dolzmann, A. and Sturm, T. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, 1997.

[3] Gao, X.S., and Chou, S.C. Solving Parametric Algebraic Systems. In Wang, D. editor, *Proc. ISSAC'92*, pages 335-341, ACM press, 1992.

[4] Greuel, G-M. and Pfister, G. *A Singular Introduction to Commutative Algebra*. Springer, 2002.

[5] Kalkbrener, M. Solving systems of algebraic equations by using Gröbner bases. In Davenport, J., editor, *EUROCAL 87*, pages 282–292. Springer, 1987.

[6] Kalkbrener, M. On the Stability of Gröbner Bases Under Specializations. *Journal of Symbolic Computation*, 24:51–58, 1997.

[7] Kapur, D. An Approach for Solving Systems of Parametric Polynomial Equations. In Saraswat and Hentenryck,V. editors, *Principles and Practice of Constraint Programming*, pages 217–244, MIT press

[8] Kreuzer, M. and Robbiano, L. *Computational Commutative Algebra 1*. Springer, 2000.

[9] Kredel, H. and Weispfenning, V. Parametric Gröbner bases for non-commutative Polynomials. In *IV. Int. Cov. Computer Algebra in Physical Research 1990*, pages 236–244. World Scientific, 1991.

[10] Kredel, H. *Solvable Polynomial Rings*. Universität Passau, Germany, 1992. Ph.D. Thesis.

[11] Möller, M. and Mora, F. New Constructive Methods in Classical Ideal Theory. *Journal of Algebra*, 100:138–178, 1986.

[12] Manubens, M. and Montes, A. Improving DISPGB algorithm using the discriminant ideal. *Journal of Symbolic Computation*, 41:1245–1263, 2006.

[13] Montes, A. A new algorithm for discussing Gröbner basis with parameters. *Journal of Symbolic Computation*, 33/1-2:183–208, 2002.

[14] Montes, A. and Recio, T., Automatic discovery of geometry theorems using minimal canonical comprehensive Gröbner system, *ADG 2006, LNAI 4869*, pages 113-138, Springer-Verlag, 2007.

[15] Nabeshima, K. A Speed-Up of the Algorithm for Computing Comprehensive Gröbner Systems. In Christopher Brown, editor, *International Symposium on Symbolic and Algebraic Computation*, pages 299–306. ACM Press, 2007.

[16] Nabeshima, K. On the Computation of Parametric Gröbner Bases for Modules and Syzygies. *Japan Journal of Industrial and Applied Mathematics*, to appear, 2009.

[17] Noro, M. and Takeshima, T. Risa/Asir- A Computer Algebra System. In Wang, P., editor, *International Symposium on Symbolic and Algebraic Computation*, pages 387–396. AMC-Press, 1992.

[18] Suzuki, A. and Sato, Y. An alternative approach to Comprehensive Gröbner bases. *Journal of Symbolic Computation*, 36/3-4:649–667, 2003.

[19] Suzuki, A. and Sato, Y. A Simple Algorithm to compute Comprehensive Gröbner Bases using Gröbner bases. In *International Symposium on Symbolic and Algebraic Computation*, pages 326–331. ACM Press, 2006.

[20] Weispfenning, V. Comprehensive Gröbner bases. *Journal of Symbolic Computation*, 14/1:1–29, 1992.

[21] Weispfenning, V. Canonical Comprehensive Gröbner bases. *Journal of Symbolic Computation*, 36/3-4:669–683, 2003.

[22] Weispfenning,V. Comprehensive Gröbner bases and regular rings. *Journal of Symbolic Computation*, 41:297–316, 2006

[23] Yokoyama, K. Stability of parametric decomposition. In Iglesias, A. and Takayama, N. editors, *International Congress on Mathematical Software,LNCS 4151*, pages 391–402. Springer Berlin / Heidelberg, 2006.

An Algorithm to Compute Parametric Standard Bases Using Algebraic Local Cohomology for Zero Dimensional Ideals (Extended Abstract)

Katsusuke Nabeshima[1], Yayoi Nakamura[2] and Shinichi Tajima[3]

[1] Graduate School of Information Science and Technology, Osaka University
Machikaneyama 1-1, Toyonaka 560-0043, Osaka, JAPAN, and
Japan Science and Technology Agency, CREST
Sanbancho, Chiyoda-ku, Tokyo, 102-0075, JAPAN.
nabeshima@math.sci.osaka-u.ac.jp

[2] School of Science and Engineering, Kinki University
3-4-1, Kowakae, Higashiosaka, Osaka, JAPAN
yayoi@math.kindai.ac.jp

[3] Faculty of Engineering, Niigata University
8050, Ninomachi, Igarashi, Nishiku, Niigata, JAPAN
tajima@ie.niigata-u.ac.jp

1 Introduction

We describe a method for computing parametric standard bases of zero-dimensional ideals with parameters. There exist algorithms and softwares for computing parametric Gröbner bases [9, 11, 14], because parametric Gröbner bases are becoming important tools for applications in several fields both inside and outside mathematics. Parametric standard bases are also useful tools for studying mathematics. However, there is no software for computing parametric standard bases, and no algorithm which guarantees the termination. In this article, we give a new algorithm for treating parametric standard bases of zero-dimensional ideals with parameters. The motivation is to study the variations of the singular locus of a parametric multivariate polynomial f over $\mathbb{C}[x_1, \ldots, x_n]$.

The theory of standard bases for ideals in power series rings was introduced in 1964 by H. Hironaka [5] on the resolution of singularities. Since then, standard bases have been extensively utilized in various fields. There are now two classical and widely used method that compute standard bases of ideals, in local rings, generated by polynomials. One method is based on Mora's tangent cone algorithm and the other is based on Lazard's homogenization technique [3, 4, 8]. For the zero-dimensional case, there is an another approach, called duality method to deal with ideals in local rings, which has also been extensively studied by several authors in the context of computer algebra[1, 6, 7]. In [18], we have adopted another classical duality, the Grothendieck duality on local residues, for treating standard bases of zero-dimensional ideals. The key ingredient in this approach is the concept of algebraic local cohomology which was introduced by A. Grothendieck in the context of Algebraic Geometry [2]. The paper [18] shows that the use of algebraic local cohomology provides an efficient method for computing standard bases. In general, algorithms for computing standard bases do not guarantee the termination. However, our algorithm [18] for computing standard bases of zero-dimensional ideals (with a singular

*Correspondence to: Graduate School of Information Science and Technology, Osaka University, Machikaneyama 1-1, Toyonaka 560-0043, Osaka, JAPAN. Tel:+81-6-6850-5551

point at the origin), guarantee the termination. Moreover, the algorithm ends up only with linear algebra.

We generalize our algorithm [18] to the case of parametric ideals for computing parametric standard bases. In the algorithm generalized, we need to algorithms for computing algebraic local cohomology classes, which are from [10, 15, 16, 17, 18]. Moreover, we need a lot of techniques for treating equations with parameters. Some of the techniques are from [11, 13, 14]. In this article, we give the rough procedure for computing parametric standard bases, and an example.

2 Rough Procedure

Due to the page restriction, we give a rough procedure for computing parametric standard bases. In order to compute algebraic local cohomology classes, we need to consider a classification of dimensions for parametric ideals, because the dimensions are changed by the values of the parameters and we are interested in only zero-dimensional ideals. For example, let $I = \langle ax, by^2+y\rangle$ be an ideal in $\mathbb{C}[x, y]$ with parameter a and b. In this case, if $a = 0, b \neq 0$, then the dimension of this ideal is clearly 1. However, if $ab \neq 0$, then the dimension of this ideal is clearly 0. As the algorithm [18] works for a zero-dimensional ideal, we need the classification of dimensions for parametric ideals. If the ideal is not zero-dimensional, we do not consider the ideal. (We consider only the case of zero-dimensional ideal, because algebraic local cohomology classes are defined by only zero-dimensional ideals. We would like to use parametric standard bases for studying singularities.) After the classification, we need to compute algebraic local cohomology classes with parameters. By the information of the algebraic local cohomology, we can easily compute parametric standard bases.

Algorithm PSD
Input: f: a polynomial in $\mathbb{C}[x_1, \ldots, x_n]$ with parameters $a_1, \ldots, a_m$ such that f has the singular point at the origin, $\succ$: a local term oder, $\succ_1$: a global term order,
Output: $\mathcal{G} = \{(S_i, G_i) | S_i \subseteq \mathbb{C}^m, G_i$ is a standard basis on S_i of $J := \langle \partial f/\partial x_1, \ldots, \partial f/\partial x_n\rangle$ w.r.t. $\succ\}$, $\mathcal{E} = \{E_i | E_i \subseteq \mathbb{C}^m, J$ is not zero-dimensional in $\succ_1$ on $E_i\}$.

1. Compute a classification of dimensions for J in $\succ_1$.
2. Compute algebraic local cohomology classes for the case of zero-dimensional ideal.
3. For each parameters' space, computer a standard basis for J by using the information of the each basis of algebraic local cohomology.

Note that, we need a **local term order** $\succ$ to compute a standard basis. In local term orders, it is difficult to classify the parameters of the Jacobian ideal of f in function of its dimension. Therefore, we need to consider another way without local term orders. If we use a global term order for computing the classification, then we can get the classification in the mean of the global term order (in the first step). If an ideal is zero dimension in the mean of global, then the ideal is zero dimension in the mean of local (at the origin). However, if an ideal is not zero dimension in the mean of global, then we can not say that is not zero dimension in the mean of local (at the origin). Therefore, the algorithm above is not complete for all zero-dimensional cases. **After the procedure, we need to check the dimension of J on $\mathcal{E}$ by other way.** In this point, we are still studying. Anyway, now in some cases, we can study singularities by using the algorithm above. (Not complete, however, in some cases, the algorithm gives us good enough information.) Remark that, in the computation, we need a lot of techniques for treating parametric equations. For

example, parametric Gröbner basis, solving a system of parametric linear equations and computations of parameters' spaces.

3 Examples

The algorithm PSD has been implemented in the computer algebra system Risa/Asir [12]. We demonstrates an example. Let $f = x^3 + x^2y^3 + ay^9 + xy^7$ in $\mathbb{C}[x, y]$ with parameters a. Our program outputs a parametric standard basis for $J = \langle \partial f/\partial x, \partial f/\partial y \rangle$ with respect to the local degree reverse lexicographical term order $\succ$ such that $y \succ x$. (I.e., $1 \succ y \succ x \succ y^2 \succ yx \succ x^2 \succ \cdots$)

```
[368] p_std(x^3+x^2*y^3+a*y^9+x*y^7,[a],[x,y],1);
non zero-dim.
[]
parametric standard bases
[[10584*a+1331],[1]]
[x^2+2/3*y^3*x+1/3*y^7,y^5*x+1667/192*y^10+1667/672*y^9+1331/2352*y^8,y^11]
[[a],[1]]
[x^2+2/3*y^3*x+1/3*y^7,y^5*x+343/16*y^12+49/8*y^11+7/4*y^10+1/2*y^9,y^13]
[[27*a+4],[1]]
[x^2+2/3*y^3*x+1/3*y^7,y^5*x+833/24*y^11+(-441/8*a+7/4)*y^10+(-63/4*a+1/2)*
y^9-9/2*a*y^8,y^12]
[[0],[18003384*a^4+4359663*a^3+178866*a^2-10648*a]]
[x^2+2/3*y^3*x+1/3*y^7,y^5*x+(-441/8*a+7/4)*y^10+(-63/4*a+1/2)*y^9-9/2*a*y^
8,y^11]
[[63*a-2],[1]]
[x^2+2/3*y^3*x+1/3*y^7,y^5*x+(-63/4*a+1/2)*y^9-9/2*a*y^8,y^11]
```

This output means the following.

- ◯ For any value of the parameter a, J is always zero-dimensional.
- ◯ If the parameters a take a value from $\mathbb{V}(a)$, then a standard basis of J w.r.t. $\succ$ is $\{x^2 + 2/3y^3x + 1/3y^7, y^5x + 1667/192y^{10} + 1667/672y^9 + 1331/2352y^8, y^{11}\}$.
- ◯ If $\mathbb{V}(27a + 4)$, then a standard basis is $\{x^2 + 2/3y^3x + 1/3y^7, y^5x + 833/24y^{11} + (-441/8a + 7/4)y^{10} + (-63/4a + 1/2)y^9 - 9/2ay^8, y^{12}\}$.
- ◯ If $\mathbb{C} \setminus \mathbb{V}(18003384a^4+4359663a^3+178866a^2-10648a)$, then $\{x^2+2/3y^3x+1/3y^7, y^5x+(-441/8a + 7/4)y^{10} + (-63/4a + 1/2)y^9 - 9/2ay^8, y^{11}\}$.
- ◯ If $\mathbb{V}(63a - 2)$, $\{x^2 + 2/3y^3x + 1/3y^7, y^5x + (-63/4a + 1/2)y^9 - 9/2ay^8, y^{11}\}$.

References and Notes

[1] Alonso, M.E., Marinari, S.M., and Mora, T. The big mother of all dualities: Möller algorithm, *Comm. in Algebra*, 31, pages 783–818, 2003.

[2] Grothendiek, A. Local Cohomology, *Lecture Notes in Math.*, 41, Springer, Berlin, 1967.

[3] Grassmann, H., Greuel, G.-M., Martin, B., Neumann, W., Pfister, G., Pohl, W., Schönemann, H. and Siebert, T. On an implementation of standard bases and syzyies in SINGULAR, *AAECC*, 7, pages 235–249, 1996.

[4] Lazard, D. Gröbner bases, Gaussian elimination, and resolution of systems of algebraic equations, *Lecture Note in Computer Science*, 162, pages 146–156, 1983.

[5] Hironaka, H. Resolution of singularities of an algebraic variety over a field of characteristic zero, *Ann. Math.*, 79, pages 109–326, 1964.

[6] Marinari, M.G., Möller H.M. and Mora, T., Gröbner bases of ideals given by dual bases, *proc. ISSAC 1991*, pages 55–63, AMC-press, 1991.

[7] Mourrain, B. Isolated points, duality and residues, *Journal of Pure and Applied Algebra*, 117 & 118, pages 469–493, 1997.

[8] Mora, T. An algorithm to compute the equations of tangent cones, *Lecture Notes in Computer Science*, 144, pages 158–165, 1982.

[9] Montes, A. A new algorithm for discussing Gröbner basis with parameters, *Journal of Symbolic Computation*, 33-1, pages 183–208, 2002.

[10] Nakamura, Y. and Tajima, S. On weighted-degrees for algebraic local cohomologies associated with semiquasihomogeneous singularities, *Advanced Studies in Pure Mathematics*, Vol. 46, pages 105-117, 2007

[11] Nabeshima, K. A Speed-Up of the Algorithm for Computing Comprehensive Gröbner Systems. In Christopher Brown, editor, *International Symposium on Symbolic and Algebraic Computation*, pages 299–306. ACM Press, 2007.

[12] Noro, M. and Takeshima, T. Risa/Asir- A Computer Algebra System. In Wang, P., editor, *International Symposium on Symbolic and Algebraic Computation*, pages 387–396. AMC-Press, 1992.

[13] Suzuki, A. and Sato, Y. An alternative approach to Comprehensive Gröbner bases. *Journal of Symbolic Computation*, 36/3-4, pages 649–667, 2003.

[14] Suzuki, A. and Sato, Y. A Simple Algorithm to compute Comprehensive Gröbner Bases using Gröbner bases. In *International Symposium on Symbolic and Algebraic Computation*, pages 326–331. ACM Press, 2006.

[15] Tajima, S. A New Look at the Local Solvanility Condition of Inhomogeneous Ordinary Differentail Equations *Publ. Res. Inst. Math. Sci., Kyoto University*, Vol. 43, N0.2, Pages 443–459, 2007.

[16] Tajima, S. and Nakamura, Y. Algebraic Local Cohomology Classes Attached to Quasi-Homogeneous Hypersurface Isolated Singularities, *Publ. Res. Inst. Math. Sci., Kyoto University*, Vol. 41, No.1, pages 1–10, 2005.

[17] Tajima, S. and Nakamura, Y. Annihilating ideals for an algebraic local cohomology class, *Journal of Symbolic Computation*, 44-5, pages 435–448. 2008

[18] Tajima, S., Nakamura, Y. and Nabeshima, K. Standard bases and algebraic local cohomology for zero dimensional ideals, *Advanced Studies in Pure Mathematics*, Vol. 56, pages 341–361, 2009

[19] Weispfenning, V. Comprehensive Gröbner bases. *Journal of Symbolic Computation*, 14-1, pages 1–29, 1992.

Discrete Mathematics and Computer Algebra System.

Masakazu Naito[1], Toshiyuki Yamauchi[1], Taishi Inoue[1], Yuuki Tomari[1],Koichiro Nishimura[1],Takuma Nakaoka[1],Soh Tatsumi[1],Ryohei Miyadera[1] , Wataru Ogasa[1] and Daisuke Minematsu[2]

[1] Kwansei Gakuin,
Uegahara 1-1-155 Nishinomiya City, Japan.
Miyadera127@aol.com

[2] Osaka University,
Toyonaka City Osaka prefecture

ABSTRACT

Here we are going to introduce a case of the application of computer algebra systems to mathematics research and its implication for the mathematics education.
We are going to study three themes. The first is the Josephus problem in both direction, the second is bitter chocolate problems that are variants of the game of Nim and the third is a new reiterative process. With the power of computer algebra system Mathematica the authors discovered sequences with self-similarity in the Josephus problem, the Sierpinski gasket in the list of previous player's positions of the game and interesting loops in a new reiterative process. The authors are members of a group of students and a mathematician who have been doing a research using Mathematica, and most of the facts that are presented in this article have been discovered by high school students. These results shows the remarkable power of computer algebra systems in education.

1 Introduction.

In this article the authors are going to present the result of the research on the Josephus Problem in both direction, the chocolate problems that are variants of the game of Nim and a new reiterative process. In the research they have made the best of Mathematica in the research. The authors are members of a group of students and a mathematician who have been doing the research of mathematics for more than 15 years. See Remark 4.2 for the results of the research in 2009.
This is a very important fact for the application of computer algebra to mathematics education.
throughout this article the authors are going to write about the method of research in remarks.

2 The Josephus problem in both direction.

In Josephus Problem we put n players in a circle, and remove them one by one according to a fixed rule. In the Josephus problem in both directions that the authors proposed we remove numbers in both directions.

*Uegahara 1-1-155 Nishinomiya City, Japan.

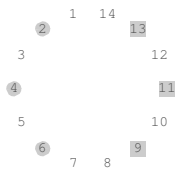
Definition 2.1. *Let n and k be natural numbers such that $k \geq 2$. We put n numbers in a circle, and two numbers are to be eliminated at the same time. These two processes of elimination go in different directions. Suppose that there are n-numbers. Then the first process of elimination starts with the 1st number and the k-th, $2k$-th, $3k$-th number, ... are to be eliminated. The second process starts with the n-th number, and the $(n-k+1)$-th, $(n-2k+1)$-th, $(n-3k+1)$-th number, ... are to be eliminated. We suppose that the first process comes first and the second process second at every stage. We denote the position of the survivor by $JI(n,k)$.*

Remark 2.1. *When students learn a problem, they usually make a new problem out of the original one. In the case of the Josephus problem they changed the way to remove numbers.*

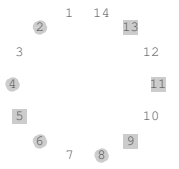
Example 2.1. *Suppose that there are $n = 14$ numbers and $k = 2$. Then the 2nd, 4th, 6th number will be eliminated by the first process. Similarly 13th, 11th, 9th number will be eliminated by the second process. Then we have Graph 2.1. Here we covered numbers eliminated by the first process and the second process with gray color disks and gray color rectangles respectively. In Graph 2.1 the last number eliminated is 9 and 6 was eliminated before it.*

Now two directions are going to overlap. The first process will eliminate 8 and the second process will eliminate 5. See Graph 2.2.

After this the first process will eliminate 12, 3, 14, and the second process will eliminate 1, 10. The number that remains is 7.



Graph 2.1.



Graph 2.2.

Example 2.2. *This Mathematica function* jose[m, k] *calculates the value of $JI(n,k)$. Note that the program is quite simple, because Mathematica has many mathematical functions that are very useful.*

```
jose[m_, k_] :=
  Block[{t, p, q, u, v, w},
   If[m == 1, 1, w = k - 1 ; t= Range[m]; p = t; q = t;
    Do[p = RotateLeft[p, w]; u = First[p]; p = Rest[p];
     q = Drop[q, Position[q, u][[1]]];
     If[Length[p] == 1, Break[],]; q = RotateRight[q, w]; v = Last[q];
      q = Drop[q, -1];
     p = Drop[p, Position[p, v][[1]]];
     If[Length[q] == 1, Break[],], {n, 1, Ceiling[m/2]}]; p[[1]]]];
```

If there is only one number, you cannot remove any number, and the number that remains is 1. Therefore jose[1, k] $= 1$ *for any natural number k.*

By using the Mathematica function jose[m, k] the authors have discovered interesting properties. They have also made recursive relations that are useful when it is necessarily

to calculate a lot of data. For recursive relations of $JI(n,k)$ see [5].
There is a very interesting fact about the function $JI(n,2)$.

We denote by $JI(\ mod\ k)$ the sequence of the least nonnegative residues of the terms of $\{JI(n,2), n=1,2,...\}$ taken modulo k. It is easy to study the sequence $JI(\ mod\ k)$ by Mathematica.

Example 2.3. *Mathematica program to produce $JI(\ mod\ k)$ is very simple. Let t be the number of the terms in the sequence, then the following program can calculate the sequence.*

```
Table[Mod[jose[m,2],k],{m,1,t}]
```

Example 2.4. *The list of the sequence $\{JI(n,2), n=1,2,3,...,\}$is*
$\{1,1,3,4,3,6,1,3,9,1,11,5,11,7,9,14,5,12,7,12,11,14,9,22,5,20,7,28,...\}$.
(1) *We are going to calculate $JI(\ mod\ 2)$.*
By the program in Example 2.3 $JI(\ mod\ 2)$ is
$\{1,1,1,0,1,0,1,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,1,1,1,1,$
$1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,...\}$.
We can find a very beautiful pattern if we arrange them as the following.
$\{1\},\{1,1\},\ \{1,0,1,0\},\ \{1,1,1,1,1,1,1,1\},\ \{1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0\},$
$\{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1\}....$
(2) *Similarly we can find a very interesting pattern in $JI(\ mod\ 4)$.*
$JI(\ mod\ 4)$ is
$\{1,1,3,0,3,2,1\},\quad \{3,1,1,3,1,3,3,1\},\quad \{2,1,0,3,0,3,2,1\},\ \{2,1,0,3,0,3,2,1\},$
$\{3,1,1,3,1,3,3,1\},\ \{3,1,1,3,1,3,3,1\},\ \{3,1,1,3,1,3,3,1\}\ ,\{3,1,1,3,1,3,3,1\}$
$,\{2,1,0,3,0,3,2,1\}\ ,\{2,1,0,3,0,3,2,1\}\ ,\{2,1,0,3,0,3,2,1\}\ ,\{2,1,0,3,0,3,2,1\}$
$,\{2,1,0,3,0,3,2,1\}\ ,\{2,1,0,3,0,3,2,1\}\ ,\{2,1,0,3,0,3,2,1\}\ ,\{2,1,0,3,0,3,2,1\}\ ...$

Remark 2.2. *The pattern in (1) of Example 2.4 is published in [5], but the pattern in (2) is a new result. By the program in Example 2.3 you can produce $JI(\ mod\ k)$ for any natural number k, and you can discover interesting patterns. The authors think that they can prove the existence of patterns for $JI(\ mod\ 2^m)$ when m is a arbitrary natural number. The pattern in Example 2.4 has been discovered by one of the authors when he has tried to find patterns by dividing numbers with various numbers. The author do not know why, but there are always several students in the group who are good at finding interesting patterns in the output data of Mathematica.*

3 A variant of the game of Nim.

In a combinatorial game there are two kinds of positions. One is a P-position, a previous-player-winning position. The other is an N-position, a next-player-winning position.
Let me explain about these positions. In the followings we use the word option to mean "choice of move". Our aim is to find all the P-positions.
It is clear that they have the following properties.

Graph 3.1.

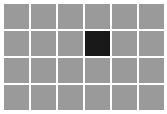
P - positions	N - Positions
Every **option** leads to an N - position	There is always at least one **option** leading to a **P -position**

Here we are going to present a bitter chocolate problem that is a variant of the game of Nim.

Definition 3.1. *Given a pieces of chocolate, where the light gray parts are sweet and the dark gray part is very bitter.Two players in turn break the chocolate (in a straight line along the grooves) and eats the piece he breaks off. The player to leave his opponent with the single bitter part is the winner.*

We are going to study this problem by using examples.

Example 3.1. *We are going to use the chocolate in Graph 3.2.*

Graph 3.2. 

This problem has been proposed in [2], and it is easy to see that this is equivalent to the traditional Nim of 4 piles.
The chocolate of Graph 3.2 has 6 columns and 4 rows, but we can study the bitter chocolate problem with arbitrary number of rows and columns. We can represent the position of the game with 4-numbers $\{x_1, x_2, x_3, x_4\}$. *For example the position of the chocolate of Graph 3.2 is* $\{1,2,2,3\}$, *since there are one row above the bitter part, two columns in the right side of it, two rows below it and three columns in the left side. These 4 coordinates are independent, i.e., you can take a number from one coordinate without affecting other coordinates.*

Definition 3.2. *Here we are going to define the nim-sum. The nim-sum is the sum (in binary) neglecting all carries from one digit to another. The nim-sum of numbers* x *and* y *is denoted by* $x \oplus y$.

Theorem 3.1. *In the game of Example 3.1* $\{x_1, x_2, x_3, x_4\}$ *is a P-position if and only if* $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$.

We omit the proof, because this is a well known fact about the game of Nim. This theorem was proved in [1].

By Theorem 3.1 the P-positions of the chocolate problem of Example 3.1 can be obtained by mathematical theory, but in many other combinatorial games it is often the case that it is difficult to get all the P-positions mathematically, and we can get the list of P-positions only by calculation of computer.
When we calculate P-positions, one of the most important tools for that is the Grundy Number.

Here we are going to define Grundy Number using the the game of Example 3.1 as an example.
First we define a very important function $Mex[\ ]$.

Definition 3.3. *The Mex of a set of nonnegative integers is the least nonnegative integer not in the set.*

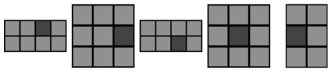
Example 3.2. $Mex[0, 1, 4, 5, 6] = 2$ *and* $Mex[1, 4, 5, 6] = 0$.

Definition 3.4. *For any position* $\boldsymbol{x}$ *we denote by* $Move[\boldsymbol{x}]$ *the set of all the positions that players can reach directly from the position* $\boldsymbol{x}$.

Example 3.3. *We are going to use the chocolate in Example 3.1 to explain about* $Move[\boldsymbol{x}]$. *Let* $\boldsymbol{x} = \{1, 1, 1, 2\}$. *Then this position is the chocolate in Graph 3.3.*

Graph 3.3. 

From this position the player can reach the positions in Graph 3.4.

Graph 3.4. 

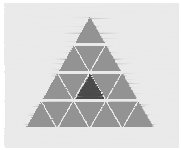
Therefore $Move[\boldsymbol{x}] = \{\{0, 1, 1, 2\}, \{1, 0, 1, 2\}, \{1, 1, 0, 2\}, \{1, 1, 1, 1\}, \{1, 1, 1, 0\}\}$.

We are going to define the Grundy Number $G(\mathbf{x})$.

Definition 3.5. *Let* P_0 *be the set of positions from which the players can have no legal option.*
For any position $\boldsymbol{x} \in P_0$ *we define* $G(\boldsymbol{x}) = 0$.
Let N_1 *be the set of positions from which the players can choose a proper option that leads to* P_0.
For any position $\boldsymbol{x} \in N_1$ *we define* $G(\boldsymbol{x}) = 1$.
For any position $\boldsymbol{x}$ *we define* $G(\boldsymbol{x})$ *recursively.*
$G(\boldsymbol{x}) = Mex[G(\boldsymbol{y}), \boldsymbol{y} \in Move[\boldsymbol{x}]]$.
For the details of the Grundy Number see [3].

By the theory of Grundy Number we know that $\mathbf{x}$ is a P-position if and only if $G(\mathbf{x}) = 0$. Therefore we can find P-positions by calculating Grundy Number $G(\mathbf{x})$.

Example 3.4. *This time we are gong to use the chocolate in Graph 3.5. This problem has been introduced by the authors.*

Graph 3.5. 

Remark 3.1. *It is fairly easy to propose a new combinatorial game, and with Mathematica you can make a program to discover the strategy to win the game. You can also find whether the new game is mathematically interesting or not.*

The problem in Graph 3.5 is different from the problem in Graph 3.2. In Graph 3.5 you can cut the chocolate in 6 ways, so it is appropriate to represent it with 6 numbers $\{x_1, x_2, x_3, x_4, x_5, x_6\}$. *We represent the position in Graph 3.5 with* $\{2,1,2,1,2,1\}$, *since there are two rows above the bitter part, one row in the right-above side, two columns in the right-below, etc.*

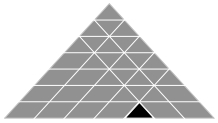
Note that these 6 coordinates are not independent, i.e., in some cases you cannot take a number from one coordinate without affecting other coordinates.
It is clear that we have 6 inequalities between these 6 coordinates.

$$x_1 \leq x_2 + x_6, x_2 \leq x_1 + x_3 + 1, x_3 \leq x_2 + x_4,$$
$$x_4 \leq x_3 + x_5 + 1, x_5 \leq x_4 + x_6, x_6 \leq x_5 + x_1.$$

We can study chocolate of any size with non-negative numbers $x_1, x_2, x_3, x_4, x_5, x_6$ that satisfy these 6 inequalities. The authors studied this chocolate problem, and calculated the list of P-positions for some cases by Mathematica. See [12]. The authors have not found any method to get P-positions mathematically.

The chocolate problem of Example 3.4 is very difficult to study mathematically, and hence the authors have made a easier version the chocolate problem. This chocolate has a very simple formula to calculate P-positions.

Example 3.5. *Suppose that you have the chocolate in Graph 3.6. In Graph 3.6 you can cut the chocolate in 3 ways, so it is appropriate to represent it with 3 numbers $\{x, y, z\}$. We represent the position in Graph 3.6 with {4, 6, 2}, since there are four columns in the left side of the bitter part, six rows above it and two columns in the right side of it. It is clear that we have an inequality between these 3 coordinates. $y \leq x + z$.*

Graph 3.6. 

Example 3.6. *We are going to find P-positions of this game in the list $\{\{x, y, z\}$, $x \leq 60$, $y \leq 60$, $z \leq 60$ and $y \leq x + z\}$ by a Mathematica program. Here we use the theory of Grundy number.*

```
ss = 60;
al
 = Flatten[Table[{a, b, c}, {a, 0, ss}, {b, 0, ss},
 {c, 0, ss}], 2];
allcases = Select[al,  (#[[1]] + #[[3]]) >= #[[2]] &];
move[z_] := Block[{p}, p = z ;
 Union[Table[{t1, Min[ (t1 + p[[3]]), p[[2]]], p[[3]]},
  {t1, 0,p[[1]] - 1}],
Table[{p[[1]], t2, p[[3]]}, {t2, 0, p[[2]] - 1}],
Table[{p[[1]], Min[ (t3 + p[[1]]), p[[2]]], t3}, {t3, 0,
      p[[3]] - 1}]
    ]
   ];
Mex[L_] := Min[Complement[Range[0, Length[L]], L]];
Gr[pos_] := Gr[pos] = Mex[Map[Gr, move[pos]]];
pposition = Select[allcases, Gr[#] == 0 &];
```

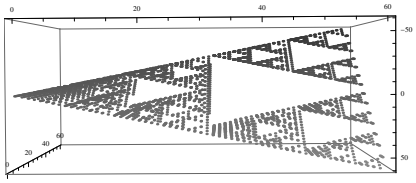
Note that it is easy to calculate the Grundy number by Mathematica, since Mathematica has many mathematical functions. By calculating Grundy numbers we can get P-positions. The list of P-positions has 3547 elements. Since this is a big list, we are going to present a part of it here.

Example 3.7. *{{0, 0, 0}, {1, 0, 1}, {1, 1, 2}, {1, 2, 3}, {1, 3, 4}, {1, 4, 5}, {1, 5, 6}, {1, 6, 7}, {1, 7, 8}, {1, 8, 9}, {1, 9, 10}, {1, 10, 11}, {1, 11, 12}, {1, 12, 13}, {1, 13, 14}, {1, 14, 15}, {1, 15, 16}, ...}*
After that we replace {a,b,c} by {a,b,c-a}, then we can get a very interesting strucuture of data. We denote this data by Data. Then
Data = {{0, 0, 0}, {1, 0, 0}, {1, 1, 1}, {1, 2, 2}, {1, 3, 3}, ...}.

Mathematica has a very powerful function for 3D graphics.
If you make a 3D graph of the list of Data, then you get the following graph.
The Mathematica program for this 3D graphics is very simple.

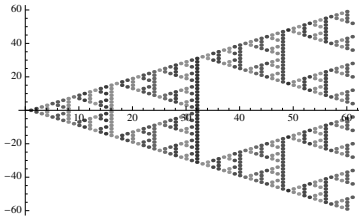
```
ListPointPlot3D[Data, PlotStyle -> PointSize[0.005]]
```

You can rotate the 3D graphics made by Mathematica, and hence the authors have rotated the 3D graph to find any interesting pattern. After some attempts the authors have discovered a pattern that looked like a Sierpinski-like gasket.



Graph 3.7.

By Graph 3.7 we know that it is important to project the 3D graph onto the plane made by the first and the third coordinates. By the projection we get Graph 3.8 .



Graph 3.8.

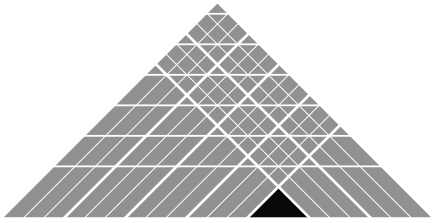
Remark 3.2. *Graph 3.8 is the Sierpinski gasket itself and this graph suggests that there must be some formulas to calculate P-positions of the game. Therefore the authors began to try to find a simple formula. After a while they discovered the following theorem.*

Theorem 3.2. (1) *The position* $\{x,\ 0,\ z\}$ *is a P-position if and only if* $x = z$.
(2) *The position* $\{x+1,\ y,\ z+1\}$ *is a P-position if and only if* $x \oplus y \oplus z = 0$.

For the proof see [7].

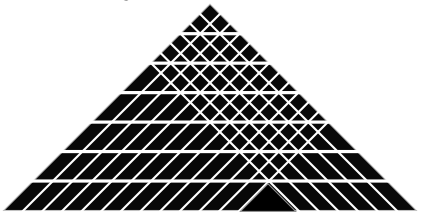
Example 3.8. *The authors are going to present two of chocolate problems with other types of inequalities.*

$3y \leq x + z$



Graph 3.9.

$y \leq \lceil \frac{x+z}{3} \rceil$



Graph 3.10.

Theorem 3.3. *The position $\{x, y, z\}$ is a P-position of the problem in Graph 3.9 if and only if $x \oplus y \oplus z = 0$ and $3y \leq x + z$.*

For the proof see [11].

Prediction 3.1. *The position $(x + 2^{\lceil log_2(y+1) \rceil} - 1, y, z + 2^{\lceil log_2(y+1) \rceil} - 1)$is a P-position of the problem in Graph 3.10 if and only if $x \oplus y \oplus z = 0$.*

Remark 3.3. *The authors predict that for an arbitrary odd number k we can prove theorems that are very similar to Theorem 3.3 and Theorem 3.1 for the chocolate problems with inequalities $ky \leq x + z$ and $y \leq \lceil \frac{x+z}{k} \rceil$ respectively. This prediction has been obtained by the calculation of Mathematica. They have never discovered any formula for the P-positions of the chocolate problem with inequalities $ky \leq x + z$ with an even number k.*

4 Reiterative Process

Next we are going to study reiterative processes. One of the most well known reiterative process is the Collatz Problem.
We are going to define Collatz function $co(n)$ for an arbitrary positive integer.

Definition 4.1. *We define Collatz fucntion by*

$$co(n) = \begin{cases} \frac{n}{2}, & (\textit{if } n \textit{ is even.}) \\ 3n + 1, & (\textit{if } n \textit{ is odd.}) \end{cases}$$

Example 4.1. *For example, if we apply Collatz function co(n) to the number 156 repeatedly, then we have*

{156, 78, 39, 118, 59, 178, 89, 268, 134, 67, 202, 101, 304, 152, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1,...}.

Remark 4.1. *Since Collatz process is a very difficult problem, students of the research group made a new reiterative process, and after that they made a Mathematica program and began the research. Reiterative processes are very good topic for students to study by making computer programs.*

The authors are going to present a new process. This is the digit factorial sum process.

Definition 4.2. *We define the dfs function by*
$dfs(n) = n_1! + \cdots n_m!$, where $\{n_1, n_2, \cdots, n_m\}$ is the list of the digits of an integer n. we start with any positive integer n, and repeatedly apply the function dfs, then we can generate a sequence of integers $\{dfs^m(n), m = 1, 2, ...\}$. We call this sequence the dfs process.

Lemma 4.1. *For any natural number k such that $k \geq 8$ we have*

$$k \times 9! < 10^{k-1}. \tag{4.1}$$

Proof. *This is an easy inequality, and hence we will omit the proof.*

Lemma 4.2. *If $m > 7 \times 9! = 2540160$, then $dfs(m) < m$.*

Proof. *Let k be the number of digits of m.*
(1) *If $k \geq 8$, then by Lemma 4.1*

$$dfs(m) \leq k \times 9! < 10^{k-1} \leq m. \tag{4.2}$$

(2) *if $k = 7$, then*

$$dfs(m) \leq 7 \times 9! = 2540160 < m. \tag{4.3}$$

Lemma 4.3. (1) *If m = 2540160, then $dsf(m) < m$.*
(2) *If $2000000 \leq m < 7 \times 9! = 2540160$, then $dfs(m) < m$.*
(3) *If $m = 1999999$, then $dfs(m) = 2177281 > m$.*

Proof. (1) *If m = 2540160, then $dfs(m) = 869$.*
(2) *If $2000000 \leq m < 7 \times 9! = 2540160$, then*

$$dfs(m) \leq 2! + 4! + 5 \times 9! = 1814426 < m. \tag{4.4}$$

(3) *If $m = 1999999$, then $dfs(m) = 2177281 > m$.*

Theorem 4.1. *For any natural number n $\{dfs^m(n)$ m = 1,2,3,...$\}$ eventually enters into one of 4 fixed points or 3 loops.*
(1) *4 fixed points are $\{1\}$,$\{2\}$,$\{145\}$, and $\{40585\}$.*
(2) *3 loops are $\{871,45361\}$, $\{872,45362\}$ and $\{169,363601,1454\}$.*

Proof. *By Lemma 4.2 and Lemma 4.3 $dfs(n)$ is decreasing for $n \leq 2 \times 10^6$, and hence we have only to study the numbers $n \leq 2 \times 10^6$ to look for loops. Therefore we can prove this theorem by calculation of Mathematica programs in Example 4.2 and Example 4.3.*

Example 4.2. *This is a Mathematica function to calculate the value of $dfs(n)$.*

```
dfs[x_]:=Apply[Plus,(IntegerDigits[x])!];
```

Example 4.3. *This Mathematica function* findloop22[n] *returns a list of numbers $\{n, p, t, q\}$. We start with the number n, then the sequence $(dfs^m)(n)$ enters into a loop when $m = p$, and $(dfs^p)(n) = t$. The length of the loop is q and $(dfs^{p+q})(n) = t = (dfs^p)(n)$.*

```
findloop22[n_]:=Block[{m=n,u,dat},dat={m};m=dfs[m];
While[!MemberQ[dat,m],dat=Append[dat,m];m=dfs[m]];
dat=Append[dat,m];u
=Flatten[Position[dat,Last[dat]]];
{n,u[[1]]-1,Last[dat],u[[2]]-u[[1]]};
```

Remark 4.2. *Throughout this article the authors have written about the method of research, but it is worth while to write more about it. Usually Dr.Miyadera who has been teaching this group for 15 years begins the research by presenting some well known problems, and asks students to make new problems after studying these problems. Students usually can propose many new problems, and make Mathematica programs for the problems with the help of Dr. Miyadera. After that they are going to study the output of the program. If the problems seem to be worth studying, they begin to study them hard. There have always been some students who are good at discovering interesting patterns in the output data of the programs. Once they discover some interesting patterns, next job is to prove them mathematically.*
As to the results of the research in 2009 by this group see [5], [6], [7], [8], [9], [10] and [11]. They have also got the 1st prize in the Canada Wide Virtual Science Fair. These are the

results in 2009, and there are a lot more results before 2009. These results show that high school students can discover many kinds of facts and theorems when they can use computer algebra systems properly.

Many teachers have used computer algebra systems in class rooms, and have made students to rediscover many facts of mathematics, and many people think that this is a good method to teach students how to be creative. The difference between the rediscovery and the discovery is very big, and high school students in this group have really discovered new facts of mathematics. The authors think that the best way to teach students how to be creative is to create something new. Therefore a proper introduction of computer algebra system into education can make a very big difference.

References and Notes

[1] C. Bouton, Nim, a game with a complete mathematical theory. Ann of math.,3. 1901-2.

[2] A.C.Robin, A poisoned chocolate problem, Problem corner, The Mathematical Gazette Volume 73 1989.

[3] M. H. Albert, R. J. Nowakowski and D. Wolfe, Lessons In Play, A K Peters.

[4] H. Matsui and T. Yamauchi, Formulas for Fibonacci-like Sequences Produced by Pascal-like triangles, Rose-Hulman Undergraduate Math Journal Vol.9.Issue 2, 2008.

[5] H. Matsui, T. Yamauchi, Soh Tatsumi,Takahumi Inoue,Masakazu Naito and Ryohei , Interesting Variants of the Josephus Problem, Kokyuroku No.1652, 2009, The Research Institute of Mathematical Science.

[6] M. Naito, S. Doro, D. Minematsu, and R. Miyadera, The Self-Similarity of the Josephus Problem and its Variants,Visual Mathematics, Volume 11, No. 2, 2009

[7] R. Miyadera and M. Naito, Combinatorial Games and Beautiful Graph produced by them, Visual Mathematics, Volume 11, No. 3, 2009

[8] H. Matsui, D. Minematsu, T. Yamauchi and R. Miyadera, Pascal-like triangles and Fibonacci-like sequences, Mathematical Gazette to appear

[9] Toshiyuki Yamauchi, Takahumi Inoue and Soh Tatsumi, Josephus Problem Under Various Moduli, the Rose-Hulman Undergraduate Mathematics Journal, Vol. 10, Issue 1, 2009

[10] H. Matsui and N. Totani, The Period and the Distribution of the Fibonacci-like Sequence Under Various Moduli, Rose-Hulman Institute of Technology, Undergraduate Math Journal, Vol.10. Issue 1, 2009.

[11] T. Yamauchi, T. Inoue and Y. Tomari, Variants of the Game of Nim that have Inequalities as Conditions, Rose-Hulman Institute of Technology, Undergraduate Math Journal, Vol.10. Issue 2, 2009.

[12] Bitter Chocolate Problem, " Material added 8 Jan 06 (Happy New Year)", MathPuzzle.com.

Spectral Decomposition and Eigenvectors of Matrices by Residue Calculus

Katsuyoshi OHARA[1*] and Shinichi TAJIMA[2]

[1] Faculty of Mathematics and Physics, Kanazawa University
Kakuma-machi, Kanazawa 920–1192, Japan
ohara@air.s.kanazawa-u.ac.jp

[2] Department of Information Engineering, Niigata University,
Igarashi 2–8050, Niigata 950–2181, Japan
tajima@ie.niigata-u.ac.jp

Abstract

A method for exactly computing spectral decomposition of diagonalizable matrices of rational numbers is developed and implemented on Risa/Asir a computer algebra system. Our idea is based on the residue calculus. As its application eigenvectors also can be computed. Moreover the method is compliant to parallel computation.

1 Introduction

In this paper we treat a method for computing spectral decomposition and eigenvectors of matrices by computer algebra. The eigenvalue problem in linear algebra is very important in both theory and application. Fast computation of characteristic polynomials and eigenvalues was studied by many peoples and implemented on computer algebra systems. However studies of computation of eigenvectors seem insufficient. Spectral decomposition of matrices is also important and strongly relates to eigenvectors. Although it can be given by contour integrals of resolvents of matrices theoretically, no computer algebra system seems to implement its calculation.

In section 2 and 3 we give new algorithms for exactly computing spectral decomposition and eigenvectors in case of diagonalizable matrices. Our idea is based on residue calculus of resolvents of matrices. Moreover it is compliant to parallel computation.

2 Spectral Decomposition of matrices

Let A be an n-dimensional square matrix of rational numbers and E the unit matrix of dimension n. The matrix-valued rational function $R(x) = (xE - A)^{-1}$ is called the *resolvent* of A. Suppose that A is diagonalizable, that is, the minimal polynomial $f(x)$ of A is square free and has m roots $\alpha_1, \ldots, \alpha_m$. Each root is an algebraic number. Let $\gamma(\lambda)$ be a cycle around an eigenvalue $\lambda = \alpha_j$ of A with the anti-clockwise direction. We put

$$P(\lambda) = \frac{1}{2\pi\sqrt{-1}} \int_{\gamma(\lambda)} R(x)\, dx. \tag{1}$$

It is well-known that $P(\lambda)$ is the projective matrix to the eigenspace associated with λ and the following relation holds:

$$E = \sum_{j=1}^{m} P(\alpha_j), \qquad A = \sum_{j=1}^{m} \alpha_j P(\alpha_j).$$

*Correspondence to: Kakuma-machi, Kanazawa 920–1192, JAPAN and +81–76–2646123

Then any column vector of $P(\lambda)$ is an eigenvector with λ. As described below, any element of $P(\lambda)$ is an algebraic number. The set $\{P(\alpha_j) \mid j = 1, \ldots, m\}$ is called the *spectral decomposition* of A.

Theorem 1. *If a matrix A is diagonalizable then the projective matrix $P(\lambda)$ has the following representation:*

$$P(\lambda) = b(\lambda)q(A, \lambda E) = b(A)q(A, \lambda E), \tag{2}$$

where $q(x, y) = (f(x) - f(y))/(x - y)$ and $b(x)$ is the inverse polynomial of the derivative $f'(x)$ in the quotient ring $\mathbf{Q}[x]/\langle f(x)\rangle$.

Proof. Clearly $q(x, y)$ is a symmetric polynomial in $\mathbf{Q}[x, y]$. From $f(A) = O$ the resolvent is written as

$$\begin{aligned} R(x) &= \frac{1}{f(x)}(f(x)E - f(A))R(x) = \frac{1}{f(x)}[(f(xE) - f(A))(xE - A)^{-1}] \\ &= \frac{1}{f(x)}q(A, xE). \end{aligned}$$

Since $f(x)$ is square free, there exist polynomials $a(x), b(x) \in \mathbf{Q}[x]$ as follows:

$$a(x)f(x) + b(x)f'(x) = \gcd(f(x), f'(x)) = 1.$$

So $b(x)$ is the inverse of $f'(x)$ in the ring $\mathbf{Q}[x]/\langle f(x)\rangle$. By the residue theorem we have

$$\begin{aligned} P(\lambda) &= \frac{1}{2\pi\sqrt{-1}}\int_{\gamma(\lambda)} R(x)dx = \frac{1}{2\pi\sqrt{-1}}\int_{\gamma(\lambda)} \frac{1}{f(x)}q(A, xE)dx \\ &= \frac{1}{2\pi\sqrt{-1}}\int_{\gamma(\lambda)} \left\{ b(x)q(A, xE)\frac{f'(x)}{f(x)} + a(x)q(A, xE) \right\} dx \\ &= b(\lambda)q(A, \lambda E). \end{aligned}$$

Clearly any element of $P(\lambda)$ lies in the field $\mathbf{Q}(\lambda)$. On the other hand it holds that

$$b(y)q(x, y) = b(x)q(y, x) - f(x)q_b(x, y) + f(y)q_b(x, y)$$

where $q_b(x, y) = (b(x) - b(y))/(x - y) \in \mathbf{Q}[x, y]$. Finally, by using $f(\lambda) = 0$ and $f(A) = O$, we have $b(\lambda)q(A, \lambda E) = b(A)q(\lambda E, A)$. □

Note that $\deg f(x) = m$. The polynomial $q(x, y) = \sum_{i=0}^{m-1} c_i(x)y^i$ is expressed by the following iteration:

$$c_{m-1}(x) = 1, \qquad c_{m-k-1}(x) = c_{m-k}(x)\, x + a_{m-k} \quad (k = 1, \ldots, m-1),$$

where $f(x) = \sum_{i=0}^{m} a_i x^i$ and $a_m = 1$. Hence we have the following algorithm for computing spectral decomposition by the second equality of the theorem.

Algorithm 1. Input: A a diagonalizable matrix, Output: $(f(x), P(\lambda))$ the pair of the minimal polynomial and the projective matrix

1. Calculation of the minimal polynomial $f(x)$ of A.
2. Calculation of the inverse $b(x) = f'(x)^{-1}$ (in $\mathbf{Q}[x]/\langle f(x)\rangle$)

3. $k \leftarrow \deg f - 1$, $C_k \leftarrow b(A)$
   for $i = k-1, \ldots, 0$; **do** $C_i \leftarrow C_{i+1}A + (\mathrm{coef}_{i+1}\, f)\, C_k$; **done**
   $P(\lambda) \leftarrow \lambda^k C_k + \cdots + \lambda C_1 + C_0$.

We implemented this algorithm on Risa/Asir a computer algebra system. By computer experiments we can see that step 3 spends the most part of computing time in Algorithm 1.

Let us discuss the complexity of step 3. Note that $\deg f(x) = m$ and $\deg b(x) \leq m-1$. In step 3 we need $2m-2$ multiplication of n-dimensional matrices of rational numbers at most. The complexity of a multiplication of matrices is $O(n^{2.376})$ by Strassen-Winograd's method. Therefore the total complexity of multiplications of step 3 is $O(mn^{2.376})$.

If $f(x)$ is reducible over $\mathbf{Q}[x]$ then we can prove the following theorem.

Theorem 2. *In addition to the hypothesis of Theorem 1, suppose that $f(x)$ can be factorized to $g(x)h(x)$ over $\mathbf{Q}[x]$. Then, for a root λ of $g(x)$,*

$$P(\lambda) = b_g(\lambda)d(\lambda)h(A)q_g(A, \lambda E)$$

Here $b_g(x)$ and $d(x)$ are the inverse polynomial of $g'(x)$ and $h(x)$ in the ring $\mathbf{Q}[x]/\langle g(x)\rangle$ respectively and $q_g(x,y) = (g(x) - g(y))/(x-y)$.

Note that, by using Theorem 2, we have a faster algorithm for computing spectral decomposition in case of square free and reducible minimal polynomials.

3 Eigenvectors

Recall that any column vector of the projective matrix $P(\lambda)$ lies in the eigenspace with λ. Since $P(\lambda) = b(\lambda)q(A, \lambda E)$ and $b(\lambda)$ is non-zero scalar, any column vector of $q(A, \lambda E)$ lies in the eigenspace with λ. That is, for computing an eigenvector with λ, we need to calculate only non-zero column vector of $q(A, \lambda E)$. However, if the minimal polynomial is irreducible, then it is known that no column vector of $P(\lambda)$ is zero in general theory of spectral decomposition.

Hence, by taking the first column vector of $q(A, \lambda E)$, we have the following algorithm in case of irreducible $f(x)$.

Algorithm 2. Input: A a diagonalizable matrix, Output: $(f(x), \boldsymbol{v}(\lambda))$ the pair of the minimal polynomial and an eigenvector

1. Calculation of the minimal polynomial $f(x)$ of A.
2. $k \leftarrow \deg f - 1$
   $\boldsymbol{e}_0 \leftarrow {}^t(1, 0, \ldots, 0)$, $\boldsymbol{u}_k \leftarrow \boldsymbol{e}_0$
   for $i = k-1, \ldots, 0$; **do** $\boldsymbol{u}_i \leftarrow A\boldsymbol{u}_{i+1} + (\mathrm{coef}_{i+1}\, f)\, \boldsymbol{e}_0$; **done**
   $\boldsymbol{v} \leftarrow \lambda^k \boldsymbol{u}_k + \cdots + \lambda \boldsymbol{u}_1 + \boldsymbol{u}_0$.

Note that in case of reducible $f(x)$ we can also have a method for computing eigenvectors by Theorem 2.

4 Parallel computation of Spectral Decomposition

Today computers with multi-processors are populated. Risa/Asir has a mechanism for parallel computation on distributed memory systems. Let us discuss parallelization of

step 3 of Algorithm 1. Let N be number of processors and suppose $N \ll n$. Any square matrix M can be divided to N blocks of row vectors. Denote by M_j the j-th block of M. Then the j-th block of $b(A)$ is written as $b(A)_j = b_0E_j + b_1A_j + \cdots + b_{m-1}A_jA^{m-2}$ for $b(x) = \sum_{k=0}^{m-1} b_k x^k$. The block $b(A)_j$ can be independently computed on each processor without communications. The iteration part of step 3 can be also divided to blocks. Therefore step 3 can be parallelizable without communications between processors except the entrance.

For example we show the elapsed time of an implementation of sequential and parallel version of Algorithm 1 on Risa/Asir. The used PC has two Intel Xeon 3.0 GHz quad core processors and 32 GB memory with Linux 2.6.18. Each element of matrices is a random integer at most 18 bits and characteristic polynomials are irreducible.

Table 1: Elapsed Time

dim. of matrix	24	32	40	48	56	64	72	80
sequential	3.20	16.50	57.06	166.00	394.70	873.82	1753.21	3259.38
4 process	1.80	8.53	24.16	72.24	144.46	314.03	592.98	1045.94

We remark that step 3 consists of multiplications of matrices. Fast algorithms are known to multiply matrices on parallel computers. (see e.g. [2]) Our method has the advantage of no communication. However, if costs of communications are ignored, the method can be more optimized by using parallel multiplication algorithms.

5 Conclusion

We developed a method for computing spectral decomposition and eigenvectors exactly in case of diagonalizable matrices. Our method is compliant to parallel computation for large size matrices.

References and Notes

[1] J. Abdeljaouted et H. Lombardi, Méthodes matricielles – Introduction à la complexité algébrique, Springer-Verlag, 2004.

[2] D. Bini and V. Y. Pan, *Polynomial and Matrix Computations, vol.1, Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[3] J. S. Frame, A simple recurrent formula for inverting a matrix (abstract), Bull. Amer. Math. Soc., **55**, p.1045., 1949.

[4] T. Kato, A Short Introduction to Perturbation Theory for Linear Operators, Springer-Verlag, 1982.

[5] S. Moritsugu and K. Kuriyama, Symbolic computation in computer algebra – eigenvalues and eigenvectors (in Japanese), Japan J. of Industrial and Applied Math., **11**, 103–120, 2001.

[6] Y. Nakamura and S. Tajima, Residue calculus with differential operators, Kyushu J. of Math., **54**, 127–138, 2000.

[7] C. Pernet, Algèbre linéarie exacte efficace: le calcul du polynôme caractéristique, PhD dissertation, University Joseph Fourier, 2006.

Characterizing the Intersection Pattern of Two Conics: A Bezoutian-Based Approach

Sylvain Petitjean[1]

[1] INRIA Nancy / LORIA, BP 239, Campus scientifique, 54506 Vandœuvre cedex, France
Sylvain.Petitjean@loria.fr

Abstract

We show how to efficiently detect the type of the intersection of two arbitrary plane conics. The characterization uses geometric predicates of bidegree $(6, 6)$ in the two input conics. While we already proved similar results recently (cf. [10]), the approach spelled out here attaches geometric meaning to the predicates and brings substantial geometric insight to the characterization problem, which was previously treated purely algebraically. A key tool in the new method is the Bezoutian.

1 Introduction

Finding simple geometric predicates for determining the intersection type and relative position of a pair of simple curved objects has recently received a lot of attention. Wang and Krasauskas [12] and Liu and Chen [8] obtained the characterization of certain positions of ellipses using techniques from computer algebra, but gave no complete classification. Etayo *et al.* [5] have completely tackled the case of two ellipses using tools from real algebraic geometry (Sturm-Habicht sequences). Systematizing those ideas and adopting a more global point of view, Briand [2] characterized the rigid isotopy classes of two arbitrary conics (i.e., essentially intersection type and which conic is inside the other when applicable). However, some of the predicates obtained have degrees that are far from being optimal.

In the 3D case, results have been scarce. Let us mention a characterization of the separation of two ellipsoids [13] and two somewhat different ways of computing the morphological type of the intersection of two quadrics, though without exhibiting easily computable predicates whose degree in the input can be simply understood [4, 11].

Contributions. In [10], we have shown how to compute predicates of bidegree at most $(6, 6)$ in the input for characterizing the relative position of two projective conics, largely improving upon previous results. Instrumental in the proof is the use of algebraic invariant theory, i.e. the study of the intrinsic properties of polynomial systems. Algebraic invariants were perceived as a bridge between geometry and algebra by the mathematicians of the 19th century (culminating with Klein's Erlangen Program, and the view of geometry as the study of the properties of a space invariant under the action of a group of transformations).

The main result proved in [10] is the construction of a specific conic (called a combinant), intrinsically attached to the pencil spanned by the two input conics, whose inertia is characteristic (in most cases) of the morphology of the intersection of the two conics. However, the problem is treated purely algebraically and the results have no obvious geometric meaning. In particular, why such inertia is characteristic of such orbit is obscure.

In this paper, we reprove essentially the same results, but using an entirely new approach which has the benefit of making perfectly clear the geometric meaning of the inertia of the combinant conic and overall bringing substantial geometric insight to the problem. The key intermediate tool we use that illuminates this interpretation is the Bezoutian.

Paper outline. The outline of the paper is as follows. In Section 2, we recall the definition of algebraic invariants, summarize some properties of pencils of conics, and give standard results on the invariants of pairs of conics. Section 3 presents the classification of pencils of conics under real projective transformations. In Section 4, we introduce a special conic pencil, to which any non-degenerate conic pencil can be reduced, and show how to associate to it a quartic polynomial whose root pattern "encodes" the base points of the pencil. Section 5 recalls the main properties of the Bezoutian of a univariate polynomial and applies them to the case of the quartic of Section 4, thus showing how the inertia of the Bezout matrix characterizes the type of the base points of the special pencil. Section 6 then relates the Bezoutian to covariants of the input conics, a result which is then extended to two arbitrary conics in Section 7. Finally, Section 8 develops an example involving conics depending on a parameter.

2 Preliminaries

Note as an initial motivation for considering algebraic invariant theory that the real projective type of the intersection of two conics is invariant by a simultaneous real projective transformation of the two conics.

Accordingly, we start by giving a brief introduction to the invariant theory of hypersurfaces and pairs of hypersurfaces under linear transformations (see [10] for a more complete overview). Let $\mathbb{K}$ be a field of characteristic zero, which will be set to $\mathbb{C}$ or $\mathbb{R}$ later. We state things here by considering the action of the general linear group $\mathrm{GL}_n(\mathbb{K})$ on homogeneous polynomials in $\mathbb{K}^n$, which descends to an action of the projective linear group $\mathrm{PGL}_n(\mathbb{K})$ on hypersurfaces in the projective space $\mathbb{P}^{n-1}(\mathbb{K})$. We then focus on the case of pairs of conics.

2.1 Invariant theory

The invariant theory of homogeneous polynomials (a.k.a. *forms*) is a central chapter of classical invariant theory. Let $\mathbf{a} = (a_{i_1 i_2 \cdots i_n})$ be elements of $\mathbb{K}$ and $\mathbf{x} = (x_1, \ldots, x_n)$ be coordinates of $\mathbb{K}^n$. Let

$$f(\mathbf{a}, \mathbf{x}) = \sum a_{i_1 i_2 \cdots i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$$

be a n-ary form (i.e. a homogeneous polynomial in n variables) of degree d on $\mathbb{K}$. Let $\mathbb{K}^{n*}$ be the vector space dual to $\mathbb{K}^n$, i.e. the space of all linear forms on $\mathbb{K}^n$. All n-ary forms of degree d form a $\mathbb{K}$-vector space which can be identified with $S_d(\mathbb{K}^{n*})$, the d-th symmetric power of $\mathbb{K}^{n*}$. It implies that $f = f(\mathbf{a}, \mathbf{x})$ can be identified to the vector $\mathbf{a}$ of its coefficients.

Let $\mathbb{K}[\mathbf{a}, \mathbf{x}]$ be the ring of polynomials in the coefficients of f and the variables $\mathbf{x}$, which is the ring of polynomials over the vector space $\Gamma = S_d(\mathbb{K}^{n*}) \oplus \mathbb{K}^n$. The action of the general linear group $\mathrm{GL}_n(\mathbb{K})$ on $\mathbb{K}^n$ induces a natural action on Γ. For each transformation P of $\mathrm{GL}_n(\mathbb{K})$, represented by a non-singular $n \times n$ matrix with coefficients in $\mathbb{K}$, the action $P : (\mathbf{a}, \mathbf{x}) \mapsto (\overline{\mathbf{a}}, \overline{\mathbf{x}})$ is defined by the equations $\mathbf{x} = P\,\overline{\mathbf{x}}, f(\mathbf{a}, \mathbf{x}) = f(\overline{\mathbf{a}}, \overline{\mathbf{x}})$. A polynomial $C \in \mathbb{K}[\mathbf{a}, \mathbf{x}]$ is a *covariant* of f under the action of $\mathrm{GL}_n(\mathbb{K})$ if

$$C(\overline{\mathbf{a}}, \overline{\mathbf{x}}) = (\det P)^w\, C(\mathbf{a}, \mathbf{x}),$$

where w is an integer. When the total degree in the variables $\mathbf{x}$ is 0, C is called an *invariant*.

In our situation, we consider not just one but two conics so we need to extend the above to a pair of forms $f_i(\mathbf{a}_i, \mathbf{x}), i = 1, 2$. As before, we consider the natural action of the group $\mathrm{GL}_n(\mathbb{K})$ on the polynomial ring in $\mathbf{x}$ and the $\mathbf{a}_i$'s. A polynomial $J \in \mathbb{K}[\mathbf{a}_1, \mathbf{a}_2, \mathbf{x}]$ is a *(joint) covariant* of the forms f_1, f_2 if $J(\overline{\mathbf{a}_1}, \overline{\mathbf{a}_2}, \overline{\mathbf{x}}) = (\det P)^w\, J(\mathbf{a}_1, \mathbf{a}_2, \mathbf{x})$, for all $P \in \mathrm{GL}_n(\mathbb{K})$.

2.2 Pencils of conics

A *conic* of the projective plane $\mathbb{P}^2(\mathbb{R})$ is two things:

- an algebraic object: an element of $\mathbb{P}(S_2(\mathbb{R}^{3^*}))$, the projectivization of the space $S_2(\mathbb{R}^{3^*})$ of real ternary quadratic forms;
- a geometric object: the zero set, in $\mathbb{P}^2(\mathbb{R})$, of the algebraic conic.

For the sake of simplicity, we will here largely forget the distinction between these different notions and only manipulate quadratic forms. Though slightly imprecise, this is done with the understanding that our results on ternary quadratic forms carry over to conics.

Attached to the quadratic form Q_S of a conic is a real 3×3 symmetric matrix S such that $Q_S = \mathbf{x}^T S\mathbf{x}$, where $\mathbf{x}$ are coordinates of $\mathbb{R}^3$. Since S is symmetric, all of its eigenvalues are real. Let σ^+ and σ^- be the numbers of positive and negative eigenvalues of S, respectively. The *inertia* of S (and Q_S) is the ordered pair (σ^+, σ^-). By Sylvester's Inertia Law, the inertia is invariant by real linear transformations, i.e. $\forall P \in \mathrm{GL}_3(\mathbb{R})$ the matrices S and $P^T SP$ have the same inertia [1, Thm. 4.38]. As is well known, the inertia of S can be easily deduced from a direct application of Descartes' Rule of Signs applied to the characteristic polynomial of S [1, Prop. 8.20].

A *pencil of conics* is a line in $\mathbb{P}(S_2(\mathbb{R}^{3^*}))$. The pencil is said to be *non-degenerate* if it contains non-singular conics. Two distinct conics of a pencil are said to generate it. All conics of a given pencil share a set of common points, called its *base points*. They are the intersection points of any two distinct conics. In the complex projective space, non-degenerate pencils of conics always have four base points, when counted with multiplicities.

Attached to the pencil generated by the conics Q_S and Q_T is a binary cubic

$$\mathcal{D} = \det(\lambda S + \mu T) = a\lambda^3 + b\lambda^2\mu + c\lambda\mu^2 + d\mu^3$$

called the *characteristic form* of the pencil. Let $\tilde{S} = \mathrm{adj}(S)$ and $\tilde{T} = \mathrm{adj}(T)$ be the adjoint matrices of S and T respectively (where adjoint means the usual transpose of the cofactor matrix). Let U be the symmetric matrix defined by

$$\mathrm{adj}(\lambda\tilde{S} + \mu\tilde{T}) = aS\lambda^2 + U\lambda\mu + dT\mu^2. \tag{1}$$

Finally let G be the Jacobian of Q_S, Q_T and Q_U. We have the following result:

Theorem 2.1 ([6, Chapter XIII]). *The coefficients of $\mathcal{D}$ are the generators of the algebra of joint invariants of a pair Q_S, Q_T of ternary quadratic forms under the action of* $\mathrm{GL}_3(\mathbb{C})$.

Q_U and G are joint covariants of Q_S and Q_T under the action of $\mathrm{GL}_3(\mathbb{C})$. *Every joint covariant of Q_S, Q_T can be expressed as an integral function of $a, b, c, d, Q_S, Q_T, Q_U, G$.*

The question examined in the rest of this paper will be the following: how to move from the above description of the invariants of *pairs* of conics under *complex* transformations to an understanding of some invariants of *pencils* of conics under *real* transformations.

3 Classification of intersections of conics

There are nine orbits of non-degenerate pencils of conics under the action of $\mathrm{PGL}_3(\mathbb{R})$, i.e. the projective linear group (cf. Levy [7]). The orbits belong to five groups, denoted **I**, **II**, **III**, **IV** and **IV** in Levy's nomenclature, corresponding to distinct morphologies over the complex numbers. Group **I** corresponds to generic intersections of two conics, i.e. four simple (real or complex) points. Fig. 1 gives a graphical representation of the intersection for each pencil orbit (representative conics inside each orbit are given in [10, Table 1]).

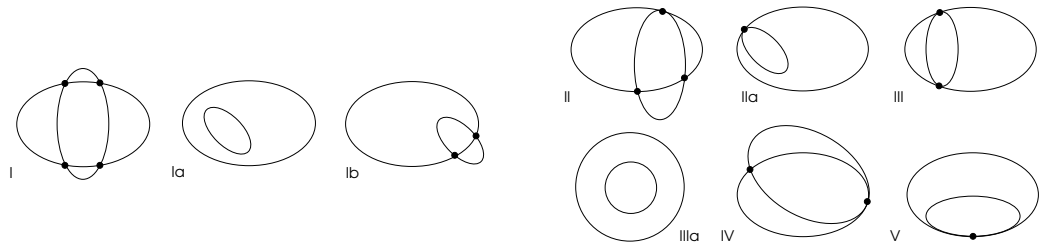


Figure 1: Morphology of the intersection for a pair of representative conics inside each pencil orbit. Left: generic cases. Right: non-generic cases.

4 A special conic pencil and associated quartic

Consider the quartic $f = a_0u^4 + 4a_1u^3v + 6a_2u^2v^2 + 4a_3uv^3 + a_4v^4$. Interpret (u, v) as homogeneous parameters defining the point $\mathbf{x} = (x, y, z) = (u^2, 2uv, v^2)$ of the conic $Q_{T'}$ whose equation is $y^2 - 4xz = 0$. Then $f = 0$ determines a set of four points on $Q_{T'}$ which are the base points of the pencil of conics

$$\lambda Q_{S'} + \mu Q_{T'} = \lambda(a_0x^2 + a_2y^2 + a_4z^2 + 2a_3yz + 2a_2xz + 2a_1xy) + \mu(y^2 - 4xz) = 0.$$

The $\mathrm{GL}_3(\mathbb{C})$-invariants of the two conics, i.e. the coefficients of $\mathcal{D} = \det(\lambda S' + \mu T')$, are

$$a' = \det S', \quad b' = a_0a_4 - 4a_1a_3 + 3a_2^2, \quad c' = 0, \quad d' = -4.$$

Up to a constant factor, the discriminant of the cubic characteristic form $\mathcal{D}$ is (see [10])

$$\Delta = b'^2c'^2 - 4a'c'^3 - 4b'^3d' - 27a'^2d'^2 + 18a'b'c'd' = 16(b'^3 - 27a'^2). \tag{2}$$

$\Delta > 0$ when $\mathcal{D}$ has three simple real roots, $\Delta < 0$ when $\mathcal{D}$ has a real root and two complex roots, and $\Delta = 0$ when $\mathcal{D}$ has a multiple root.

As is easy to observe, a linear transformation P of (u, v) induces a linear transformation P' of $\mathbf{x} = (u^2, 2uv, v^2)$, with $\det P' = (\det P)^3$. Under the transformation P', S' and T' transform to S and T respectively, with $S = P'^T S' P', T = P'^T T' P'$. Thus

$$\overline{\mathcal{D}} = \det(\lambda S + \mu T) = (\det P')^2 \det(\lambda S' + \mu T') = (\det P)^6 \mathcal{D},$$

which implies that a' and b' are $\mathrm{GL}_2(\mathbb{C})$-invariants of f. One may recognize the usual invariants of a quartic, usually denoted $J = a'$ and $I = b'$. Also, in view of the well-known fact that the intersection of two conics is generic exactly when $\mathcal{D}$ has no multiple root and the correspondence between roots of f and intersection points of the two conics, we conclude from Equation (2) that the discriminant of a quartic has the form $I^3 - 27J^2$.

Similarly, one can prove that the intersection with $Q_{T'}$ of any covariant curve of $Q_{S'}$ and $Q_{T'}$ forms a set of points determined by the vanishing of a covariant of f. Consider for instance the Hessian covariant of f:

$$g_4 = \frac{1}{144} \begin{vmatrix} \frac{\partial^2 f}{\partial u^2} & \frac{\partial^2 f}{\partial u \partial v} \\ \frac{\partial^2 f}{\partial u \partial v} & \frac{\partial^2 f}{\partial v^2} \end{vmatrix} = (a_0a_2 - a_1^2)u^4 + 2(a_0a_3 - a_1a_2)u^3v + \cdots$$

Let $Q_{U'}$ be the quadratic covariant of $Q_{S'}$ and $Q_{T'}$, as in Eq. (1). It can be observed that the intersection of $Q_{U'}$ and $Q_{T'}$ forms a set of points determined by the vanishing of g_4. More precisely, plugging $\mathbf{x} = (u^2, 2uv, v^2)$ into $Q_{U'}$, we show that:

$$g_4 = -2^{-2}\, \mathbf{x}^T U' \mathbf{x}. \tag{3}$$

The one-to-one correspondence between real (resp. complex) roots of f and real (resp. complex) intersection points of the conics $Q_{S'}$ and $Q_{T'}$ induces a correspondence at the level of orbits, which we can essentially read on Figure 1: f has only simple roots corresponds to orbit group **I**, with four real roots being real orbit **I**, four complex roots being real orbit **Ia**, etc.; f has two double roots implies orbit group **III**, with correspondence to real orbit **III** if the roots are real and **IIIa** if the roots are complex, etc.

5 Projective Bezoutian

Let us now review a standard way of "encoding" the root pattern of a polynomial. We recall here the definition of a (projective) Bezoutian, summarize its main properties and show how they apply to the case of binary quartics.

5.1 Bezout matrix

Let $\mathbf{u}_0 = (u_0, v_0), \mathbf{u}_1 = (u_1, v_1)$ be pairs of variables. If g, h are real homogeneous polynomials in u, v of degree m, then their (projective) *Bezoutian* is defined to be

$$\mathcal{B}(g,h) = \frac{g(u_0,v_0)\,h(u_1,v_1) - g(u_1,v_1)\,h(u_0,v_0)}{\omega}.$$

with $\omega = u_0 v_1 - u_1 v_0$. The Bezoutian is a polynomial, since the denominator divides the numerator. It is homogeneous in both pairs of variables $\mathbf{u}_0$ and $\mathbf{u}_1$ separately, and has degree $m-1$ in both. Write $\mathcal{B}(g,h) = \sum_{i,j=0}^{m-1} b_{ij} u_0^{m-i} v_0^i u_1^{m-j} v_1^j$. The *Bezout matrix* is the real symmetric $m \times m$ matrix defined as $B(g,h) = (b_{ij})_{i,j=0}^{m-1}$. Note that by Sylvester's Inertia Law, the inertia of $B(g,h)$ does not depend on the basis in which it is written. The Bezout matrix enjoys the following remarkable property, known as the Jacobi-Darboux Theorem: $\dim(\ker B(g,h))$ is equal to the degree of the gcd of g and h [1, Thm. 9.4].

Let now $f(u,v)$ be a real homogeneous polynomial of degree d having r (resp. $2c$) distinct real roots (resp. complex roots). Let $g = f_v, h = f_u$ so that $B(f) := B(g,h)$ is a $(d-1) \times (d-1)$ real symmetric matrix, which we call the Bezout matrix of f. Assume $B(f)$ has p_+, p_- positive and negative eigenvalues, respectively. Denoting by $\mathrm{rk}(B(f))$ the rank of $B(f)$, we have

$$\dim(\ker B(f)) = d - 1 - \mathrm{rk}(B(f)), \quad \deg(\gcd(f_u, f_v)) = d - r - 2c.$$

In view of the Jacobi-Darboux Theorem, we conclude from $\mathrm{rk}(B(f)) = p_+ + p_-$ that

$$p_+ + p_- = r + 2c - 1. \tag{4}$$

Another important result relates the signature of $B(f)$ to the number of real roots r of f (cf. [1, Cor. 9.9] adapted to the projective formulation of the Bezout matrix):

$$p_+ - p_- = r - 1. \tag{5}$$

Combining (4) and (5), we have $r = p_+ - p_- + 1$ and $c = p_-$, or conversely for the inertia

$$\mathrm{In}(B(f)) = (p_+, p_-) = (r + c - 1, c). \tag{6}$$

Remark 5.1. The Bezout matrix of f is non-singular exactly when $\mathrm{rk}(B(f)) = d - 1$. In view of the above, it implies that $r + 2c = d$, i.e. f has only simple roots. The determinant of $B(f)$ is therefore a constant multiple of the discriminant of f.

5.2 Application to quartics

Let us now return to the special conic pencil and its associated quartic introduced in Section 4. In view of the correspondence between roots of f and intersection points of the conics $Q_{S'}$ and $Q_{T'}$, the pair (r, c) of real and complex roots of f characterizes the orbit in which the special conic pencil lies (again check Figure 1), with one notable exception: the pair $(2, 0)$ corresponds both to two real double points (orbit III) and a triple point and a simple point (orbit IV). In turn, in view of (6), the inertia of $B(f)$ characterizes the pencil orbit, with the same exception concerning inertia $(1, 0)$:

- $B(f)$ has inertia $(3, 0)$: orbit I;
- $B(f)$ has inertia $(1, 2)$: orbit Ia;
- $B(f)$ has inertia $(2, 1)$: orbit Ib;
- $B(f)$ has inertia $(2, 0)$: orbit II;
- $B(f)$ has inertia $(1, 1)$: orbit IIa;
- $B(f)$ has inertia $(1, 0)$: orbits III or IV;
- $B(f)$ has inertia $(0, 1)$: orbit IIIa;
- $B(f)$ has inertia $(0, 0)$: orbit V.

To complete the characterization, we now need a way to distinguish between orbits III and IV. Let us generate an example for each orbit:

- orbit III: take $f = 6u^2v^2$, i.e. $a_2 = 1$ and $a_0 = a_1 = a_3 = a_4 = 0$. This leads to $\mathcal{D} = -(\lambda + \mu)(\lambda - 2\mu)^2$, so $\mathcal{D}$ has a double root and a simple real root.
- orbit IV: take $f = 4u^3v$, i.e. $a_1 = 1$ and $a_0 = a_2 = a_3 = a_4 = 0$. This leads to $\mathcal{D} = -4\mu^3$ for the characteristic form, so $\mathcal{D}$ has a triple real root.

Note that the root pattern of $\mathcal{D}$ does not change within an orbit. Indeed, a simultaneous linear transformation of the conics $Q_{S'}$ and $Q_{T'}$ and a linear reparameterization of the pencil variable (λ, μ) do not affect the nature of the roots of $\mathcal{D}$. We can therefore conclude in view of the following lemma:

Lemma 5.2 (Hesse, cf. [9, Prop. 2.23]). *A binary form of degree n has vanishing Hessian if and only if it is the n^{th} power of a linear form.*

So let

$$g_3 = \frac{1}{4}\begin{vmatrix} \frac{\partial^2 \mathcal{D}}{\partial \lambda^2} & \frac{\partial^2 \mathcal{D}}{\partial \lambda \partial \mu} \\ \frac{\partial^2 \mathcal{D}}{\partial \lambda \partial \mu} & \frac{\partial^2 \mathcal{D}}{\partial \mu^2} \end{vmatrix} = (3a'c' - b'^2)\lambda^2 + (9a'd' - b'c')\lambda\mu + (3b'd' - c'^2)\mu^2.$$

According to Lemma 5.2, $g_3 \equiv 0$ on orbit IV and $g_3 \not\equiv 0$ on orbit III.

6 On the Bezoutian as a sum of polarized transvectants

In the previous section, we proved how to discriminate between the different orbits (and therefore intersection types) when the pencil has the special form given in Section 4. How do we go from that to a similar result concerning an arbitrary pencil of conics? To achieve this, we need to better understand the invariant theory of the Bezoutian, which we do here in light of results due to Chipalkatti [3].

6.1 Transvectants

Let us first introduce a well-known tool in the invariant theory of binary forms, transvection, which is based on an invariant differential operator originally introduced by Cayley. Let us use two independent sets of variables $\mathbf{u}_\alpha = (u_\alpha, v_\alpha)$, $\mathbf{u}_\beta = (u_\beta, v_\beta) \in \mathbb{C}^2$. The second-order differential operator $\Omega_{\alpha\beta} = \frac{\partial^2}{\partial u_\alpha \partial v_\beta} - \frac{\partial^2}{\partial u_\beta \partial v_\alpha}$ is known as the Cayley Ω-*process* with respect to the variables (u_α, v_α) and (u_β, v_β). Below we use the "trace" notation tr to denote the substitution $u = u_\alpha = u_\beta, v = v_\alpha = v_\beta$.

Let $f_1(u, v)$ and $f_2(u, v)$ be two binary forms of degree e and f respectively. The k-*th transvectant* of f_1 and f_2 is the function

$$(f_1, f_2)_k = \frac{(e-k)!\,(f-k)!}{e!\,f!}\,\mathrm{tr}\,(\Omega_{\alpha\beta})^k[f_1(u_\alpha, v_\alpha) f_2(u_\beta, v_\beta)].$$

Then $(f_1, f_2)_k$ is a joint covariant of f_1 and f_2 [9, Theorem 5.4]. Note that $\Omega_{\beta\alpha} = -\Omega_{\alpha\beta}$ and $(f_1, f_2)_k = (-1)^k (f_2, f_1)_k$. Note also that $(f_1, f_2)_k = 0$ when $2k > e + f$.

For instance, if f is the general quartic of Section 4, then

$$g_4 = 2^{-1}(f, f)_2 = (a_0 a_2 - a_1^2)u^4 + 2(a_0 a_3 - a_1 a_2)u^3 v + \cdots \tag{7}$$

is the Hessian of f, i.e. the simplest non-trivial non-constant covariant of f, and

$$I = 2^{-1}(f, f)_4 = a_0 a_4 - 4a_1 a_3 + 3a_2^2 \tag{8}$$

is an invariant of f we have already crossed in Section 4.

Let us now prove a technical lemma which we shall need in the following section.

Lemma 6.1. *Let $f(u, v)$ be a binary form of degree d. Then $(f, f)_{2k} = \frac{2}{d^2}(f_u, f_v)_{2k-1}$.*

Proof. We have:

$$\begin{aligned}
\mathrm{tr}\,(\Omega_{\alpha\beta})^r[f(u_\alpha, v_\alpha) f(u_\beta, v_\beta)] &= \mathrm{tr}\,(\Omega_{\alpha\beta})^{r-1}[f_u(u_\alpha, v_\alpha) f_v(u_\beta, v_\beta) - f_u(u_\beta, v_\beta) f_v(u_\alpha, v_\alpha)],\\
&= \mathrm{tr}\,(\Omega_{\alpha\beta})^{r-1}[f_u(u_\alpha, v_\alpha) f_v(u_\beta, v_\beta)] - \mathrm{tr}\,(\Omega_{\alpha\beta})^{r-1}[f_u(u_\beta, v_\beta) f_v(u_\alpha, v_\alpha)],\\
&= \mathrm{tr}\,(\Omega_{\alpha\beta})^{r-1}[f_u(u_\alpha, v_\alpha) f_v(u_\beta, v_\beta)] + (-1)^r\,\mathrm{tr}\,(\Omega_{\beta\alpha})^{r-1}[f_u(u_\beta, v_\beta) f_v(u_\alpha, v_\alpha)],\\
&= (1 + (-1)^r)\,\mathrm{tr}\,(\Omega_{\alpha\beta})^{r-1}[f_u(u_\alpha, v_\alpha) f_v(u_\beta, v_\beta)].
\end{aligned}$$

The result follows in view of the definitions of $(f, f)_{2k}$ and $(f_u, f_v)_{2k-1}$. □

6.2 Polarization

Let ∂_0^1 denote the *polarization* operator $\partial_0^1 = u_1 \frac{\partial}{\partial u_0} + v_1 \frac{\partial}{\partial v_0}$. If $h(u, v)$ is a form of degree e, then define its k-*th polar* to be $h^{\langle k \rangle} = \frac{(e-k)!}{e!}(\partial_0^1)^k h(u_0, v_0)$, which is a form of degree $e - k$ in (u_0, v_0) and k in (u_1, v_1). If e is even, denote $h^{\langle e/2 \rangle}$ by $h^{\mathfrak{p}}$. Denote the "inverse" process of *restitution*, i.e. substitution of (u_0, v_0) and (u_1, v_1) by (u, v) in a form $g(u_0, v_0; u_1, v_1)$, by the symbol $g_{\mathfrak{p}}$. Note that $(h^{\mathfrak{p}})_{\mathfrak{p}} = h$.

Chipalkatti [3] gives a Taylor expansion of the Bezoutian of two forms of the same degree in terms of polarized transvectants.

Proposition 6.2 ([3, Prop. 2.4]). *Let g, h be binary forms of degree m and let c_r be the constants*

$$c_r = 2\tbinom{m}{2r+1}^2 \tbinom{2m-2r}{2r+1}^{-1}, \quad 0 \leqslant r \leqslant \left\lfloor \tfrac{m-1}{2} \right\rfloor.$$

Then, with the notations of Section 5.1, $\mathcal{B}(g, h) = \sum_{r \geqslant 0} c_r\, \omega^{2r}(g, h)^{\mathfrak{p}}_{2r+1}$.

6.3 Back to conics

Let us now apply the results of the previous sections to the Bezoutian of the quartic f, i.e. $\mathcal{B}(f) = \mathcal{B}(f_v, f_u)$. First note that when g, h have degree 3, Proposition 6.2 tells us that

$$\mathcal{B}(g,h) = 3\,T_1^{\mathfrak{p}} + 2^{-1}\omega^2 T_3^{\mathfrak{p}},$$

with $T_1 = (g,h)_1$ and $T_3 = (g,h)_3$. Assume now $g = f_v, h = f_u$. By Lemma 6.1 and Equations (7) and (8)

$$T_1 = (f_v, f_u)_1 = -8(f,f)_2 = -16\,g_4, \quad T_3 = (f_v, f_u)_3 = -8(f,f)_4 = -16\,I = -16\,b'.$$

T_3 is a constant so $T_3^{\mathfrak{p}} = T_3$, resulting in $\mathcal{B}(f) = -8\,(6\,g_4^{\mathfrak{p}} + \omega^2 b')$.

Let $\mathbf{p} = (u_0^2, 2u_0v_0, v_0^2)$ and $\mathbf{q} = (u_1^2, 2u_1v_1, v_1^2)$. Note first that $\omega^2 = (u_0v_1 - u_1v_0)^2 = -2^{-1}\,\mathbf{p}^T T'\mathbf{q}$. In view of (3) and the fact that $(g_4^{\mathfrak{p}})_{\mathfrak{p}} = g_4$, $g_4^{\mathfrak{p}}$ has the form

$$g_4^{\mathfrak{p}} = -2^{-2}\,\mathbf{p}^T U'\mathbf{q} + \kappa\,\mathbf{p}^T T'\mathbf{q},$$

where κ is a constant. A direct computation gives $\kappa = b'/6$. Overall, we get

$$\mathcal{B}(f) = 4\mathbf{p}^T(3U' - b'T')\mathbf{q}. \tag{9}$$

Remark 6.3. As a followup to Remark 5.1, the determinant of the matrix $3U' - b'T'$ is a multiple of the discriminant of the quartic f, itself a multiple of the discriminant Δ of the cubic $\det(\lambda S' + \mu T')$. A straightforward calculation shows that $\det(3U' - b'T') = \Delta$.

7 General conic pencils

We are now ready to examine the case of general pencils of conics. We first recall the following technical lemma, whose proof is given in [10]:

Lemma 7.1. *Let (Q_S, Q_T) be a general pair of conics, with quadratic covariant Q_U, and let $\det(\lambda S + \mu T) = a\lambda^3 + b\lambda^2\mu + c\lambda\mu^2 + d\mu^3$. Then the matrix of Q_U satisfies $U = cS - S\tilde{T}S = bT - T\tilde{S}T$.*

Note first that any non-degenerate pencil spanned by the conics Q_S and Q_T is projectively equivalent to a pencil of the special form given in Section 4, i.e. the pair Q_S, Q_T can be transformed to $Q_{S'}, Q_{T'}$ by an appropriate linear substitution $(Q_S, Q_T) \mapsto (\alpha Q_S + \beta Q_T, \gamma Q_S + \delta Q_T), \alpha\delta - \beta\gamma \neq 0$ and a simultaneous linear transformation of Q_S and Q_T. Indeed, a non-degenerate pencil has at least one non-singular member which can be mapped that way to the conic $Q_{T'}$, the conic $Q_{S'}$ being completely general.

Assume then we replace our pair of initial conics $(Q_{S'}, Q_{T'})$ by $(Q_S, Q_T) = (Q_{S'} + pQ_{T'}, Q_{T'})$, where p is some real parameter. Plugging the parameterization of $Q_T = Q_{T'}$ in Q_S will yield the same homogeneous polynomial of f and therefore the same Bezoutian $\mathcal{B}(f)$. The question arises: does the expression of $\mathcal{B}(f)$ in terms of invariants/covariants of the two conics change? First note that

$$\det(\lambda S + \mu T) = \det(\lambda S' + (p\lambda + \mu)T') = a'\lambda^3 + b'\lambda^2(p\lambda + \mu) + c'\lambda(p\lambda + \mu)^2 + d'(p\lambda + \mu)^3,$$

from which we see that the coefficients of $\det(\lambda S + \mu T)$ are $a = a' + pb' + p^2c' + p^3d', b = b' + 2pc' + 3p^2d', c = c' + 3pd', d = d'$. Since $c' = 0$, we have $c = 3pd$ and $b' = b - 3p^2d$.

Applying Lemma 7.1, we have $U = cS - S\tilde{T}S$ and

$$U' = c'S' - S'\tilde{T}'S' = -(S - pT)\tilde{T}(S - pT) = -S\tilde{T}S + 2pdS - p^2dT,$$

so that $U' = U - pdS - p^2dT$. Therefore the matrix of Equation (9) transforms as

$$3U' - b'T' = 3U - 3pdS - 3p^2dT - (b - 3p^2d)T = 3U - bT - 3pdS = 3U - bT - cS.$$

In other words, the matrix $\Lambda = 3U - bT - cS = 3U' - b'T' - c'S'$ is independent of the parameter p. It can be similarly proved that Λ does not depend on parameter q when $(Q_{S'}, Q_{T'})$ is replaced by $(Q_S, Q_T) = (Q_{S'}, Q_{T'} + qQ_{S'})$. In other words, Λ is invariant by a linear reparameterization of the pencil variable (λ, μ).

In addition, Q_S, Q_T, Q_U are known covariants of the pair of conics (Q_S, Q_T) (cf. Theorem 2.1), implying overall that $3Q_U - bQ_T - cQ_S$ is a covariant not just of the pair of conics but also of the underlying pencil. Sylvester called such an object a *combinant*.

Finally, note that the Hessian g_3 used in Section 5.2 is a covariant of the characteristic form $\mathcal{D} = \det(\lambda S + \mu T)$ and therefore also a combinant of the pencil.

Overall, we have proved the following (a slight restatement of the main result of [10]):

Theorem 7.2. *The characterization of pencil orbits of Section 5.2 using the inertia of $\Lambda = 3U - bT - cS$ (which replaces $B(f)$) and the vanishing/non-vanishing of g_3 carries over to the case of arbitrary non-degenerate conic pencils.*

8 Example

We consider the following pair of conics:

$$Q_S : x^2 - 4txz + ty^2 + z^2 = 0, \quad Q_T : x^2 + 2xy - 2yz - 3z^2 = 0.$$

Q_S is a function of a parameter t (time, say) and we wish to know for which values of this parameter the two conics have empty intersection. In other words, in view of Figure 1, we want to know when the pencil generated by Q_S and Q_T falls in Orbit Ia or IIIa.

The joint invariants of the pair of conics are $a = t - 4t^3, b = -2t, c = -2 + t, d = 2$. We in turn compute U using Lemma 7.1, and then Λ:

$$\Lambda = \begin{pmatrix} -1-8t+12t^2 & -7t+6t^2 & 3-4t+8t^2 \\ -7t+6t^2 & -4t+11t^2 & -5t+18t^2 \\ 3-4t+8t^2 & -5t+18t^2 & -1-16t+12t^2 \end{pmatrix}.$$

The inertia of Λ is found by inspection of its characteristic polynomial:

$$\det(\ell I - \Lambda) = \ell^3 - \operatorname{tr}(\Lambda)\ell^2 + \gamma(\Lambda)\ell - \Delta.$$

We find $\Delta = 4t(8 + t - 32t^2 + 120t^3 + 48t^4 - 428t^5), \gamma(\Lambda) = -8 + 56t + 40t^2 - 320t^3 - 16t^4$ and $\operatorname{tr}(\Lambda) = -2 - 28t + 35t^2$.

To be in orbit Ia, we must first have $\Delta > 0$, implying $t \in (-0.45, -0.44)$ or $t \in (0, 0.56)$. On each of these intervals, the morphology of the intersection does not change (otherwise, we would by continuity go through a singular intersection, corresponding to a zero of Δ). The inertia of Λ is therefore constant on each interval. It is thus sufficient to evaluate it for an arbitrary value of t inside each interval. For the first interval, we evaluate for instance at $t = -445/1000 = -89/200$ and find that the inertia of Λ is $(3, 0)$, so we are in orbit I and the intersection is not empty according to the results of Section 5.2. For the second

interval, we evaluate at $t = 1/2$ and find that the inertia of Λ is $(1, 2)$, so we are in orbit **Ia** and the intersection is empty.

To be in orbit **IIIa**, corresponding to Λ having inertia $(0, 1)$, implies in particular that Δ and $\gamma(\Lambda)$ vanish simultaneously. A little calculation shows that the resultant of these two polynomials is not zero, and therefore this case can not happen.

Overall, the intersection of the two conics is empty exactly when $t \in (0, 0.56)$.

References and Notes

[1] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 2003. Second edition.

[2] E. Briand. Equations, inequations and inequalities characterizing the configurations of two real projective conics. *Applicable Algebra in Engineering, Communication and Computing*, 18(1-2):21–52, 2007.

[3] J. Chipalkatti. On the invariant theory of the Bezoutian. *Beiträge zur Algebra und Geometrie*, 47(2):397–418, 2006.

[4] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Near-optimal parameterization of the intersection of quadrics: II. A classification of pencils. *Journal of Symbolic Computation*, 43(3):192–215, 2008.

[5] F. Etayo, L. González-Vega, and N. del Rio. A new approach to characterizing the relative position of two ellipses depending on one parameter. *Computer Aided Geometric Design*, 23(4):324–350, 2006.

[6] J.H. Grace and A. Young. *The Algebra of Invariants*. Cambridge University Press, 1903.

[7] H. Levy. *Projective and Related Geometries*. The Macmillan Co., New York, 1964.

[8] Y. Liu and F.-L. Chen. Algebraic conditions for classifying the positional relationships between two conics and their applications. *J. Comput. Sci. Technol.*, 19(5):665–673, 2004.

[9] P.J. Olver. *Classical Invariant Theory*. Cambridge University Press, 1999.

[10] S. Petitjean. Invariant-based characterization of the relative position of two projective conics. In I. Emiris, F. Sottile, and T. Theobald, editors, *Nonlinear Computational Geometry*, volume 151 of *The IMA Volumes in Mathematics and its Applications*. Springer-Verlag, 2009. To appear.

[11] C. Tu, W. Wang, B. Mourrain, and J. Wang. Using signature sequences to classify intersection curves of two quadrics. *Computer Aided Geometric Design*, 26(3):317–335, 2009.

[12] W. Wang and R. Krasauskas. Interference analysis of conics and quadrics. In *Topics in Algebraic Geometry and Geometric Modeling*, volume 334 of *Contemp. Math.*, pages 25–36. Amer. Math. Soc., 2003.

[13] W. Wang, J. Wang, and M.-S. Kim. An algebraic condition for the separation of two ellipsoids. *Computer Aided Geometric Design*, 18(6):531–539, 2001.

Certified Complex Root Isolation via Adaptive Root Separation Bounds

MICHAEL SAGRALOFF[1*], MICHAEL KERBER[2], MICHAEL HEMMER[3]

[1,2,3] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[msagralo|mkerber|hemmer]@mpi-inf.mpg.de

Abstract

We address the problem of *root isolation* for polynomial systems: for an affine, zero-dimensional polynomial system of N equations in N variables, we describe an algorithm to encapsulate all complex solutions into disjoint regions, each containing precisely one solution (called *isolating regions*). Our approach also computes the multiplicity of each solution. The main novelty is a new approach to certify that a set of computed regions is indeed isolating. It is based on an adaptive root separation bound obtained from combining information about the approximate location of roots and resultant calculus. Here we use simple subdivision method to determine the number of roots within certain regions. The resultant calculus only takes place over prime fields to avoid the disadvantageous coefficient growth in symbolic methods, without sacrificing the exactness of the output. The presented approach is complete for uni- and bivariate systems, and in general applies in higher dimensions as well, possibly after a coordinate change.

1 Introduction

Finding the roots of a zero dimensional polynomial system is a fundamental problem of numerous applications spread over several important areas, such as algebraic geometry, computer graphics and computer aided geometric design. In particular, the design of robust and certified algorithms demands for efficient methods that determine isolating regions for all roots of polynomial systems. Such methods should also be capable to handle non-simple roots.

This work is driven by the question: How can fast but unreliable root solving techniques be combined with symbolic computations in an efficient way, such that the overall result can be certified? We see our main contribution in providing a novel certification scheme in this context. Its main idea is to follow two threads of computation in parallel. Both threads only deliver incomplete information, but their combination is sufficient to certify the result of the method.

The first thread is inspired by elimination methods such as *Multivariate resultants* and *Groebner bases*. Both are well-studied tools to obtain the solution set of a system with respect to a projection direction. However, both methods lead to polynomials with very large bitsizes (for intermediate results as well as for the final result), which causes a severe drawback regarding the performance. Therefore, our method computes the multivariate resultant (with some hidden variable) only in several prime fields $\mathbb{Z}_p$ and completely avoids Chinese Remaindering. In particular, all symbolic computations are performed using single precision arithmetic [4]. This method yields a lower bound on the number of projected solutions. Although this bound is very likely to match the exact number in practice, this cannot be certified without further knowledge.

The second thread follows an *Exclusion and Subdivision method*. It keeps on subdividing regions that may contain solutions (from now on, we call such connected regions *clusters*), whereas regions that doubtlessly do not contain a solution are discarded. As a simple exclusion method we use interval arithmetic. Usually, this is combined with a criterion to ensure that a cluster contains

*Correspondence to: Michael Sagraloff, Max-Planck-Institut für Informatik, Departement 1: Algorithms and Complexity, Campus E1 4, 66123 Saarbrücken, Germany; Tel: +49 681 9325 106, Fax: +49 681 9325 199

precisely one simple root, but these criteria mostly fail in the presence of multiple roots. Instead, we introduce two new methods based on homotopy arguments. The first ensures the presence of at least one root inside a cluster, the second sums up the multiplicities of the roots inside a cluster. Clearly, this cannot suffice to ensure that a cluster is isolating.

Our algorithm merges both building blocks to certify that clusters are isolating. For simplicity, we sketch the certification idea only for a univariate polynomial f (the general case is discussed in Section 2). Two real values are computed: LB, obtained by the modular symbolic computation, and UB, obtained by the distances between clusters, diameters of clusters, and their multiplicities. If the clusters are not isolating, LB (UB) defines a lower (upper) bound on the absolute value of the first non-vanishing subresultant coefficient* $\mathrm{sres}_k(f,f')$, which is essentially the product of squared root distances of f. During the certification, $LB \to \infty$ and $UB \to 0$. Once LB becomes larger than UB, it is proven by contradiction that the clusters are isolating.

We see a strength of our certification method in its adaptiveness to the concrete instance: The quality of the bounds LB and in particular UB are mainly determined by the size of $|\mathrm{sres}_k(f,f')|$, small values for it lead to faster certification. This adaptive behavior is a clear advantage compared to theoretical a priori bounds on $\mathrm{sres}_k(f,f')$ or on the separation of distinct roots which have to assume the worst-case.

Our new certification approach is embedded within a complete algorithmic description that takes a zero-dimensional system as input, and starts the subdivision on a sufficiently large bounding box. We decided for this setup for the sake of a comprehensive description, although there is no need to restrict to the proposed subdivision strategy. For instance, we propose that for an efficient realization of our ideas, a fast numerical solver (e.g, based on homotopy methods; see the section on related work) is first applied to approximate the solutions of the system, and our certification process comes in afterwards only to validate the outcome of the solver. We sketch the workflow of such a hybrid solver in the conclusion.

The presented method is complete for 2×2-systems and applies, in general, also to higher dimensional systems. It requires the multivariate resultant to be expressible by a Macaulay matrix. In unfortunate cases, this is not possible, even under projective transformations.

Related Work. Since polynomial root solving is such an important problem in several fields, plenty of distinct approaches exist and many textbooks are dedicated to this subject. See, for instance, [9, 22, 27, 29] for introductions to symbolic approaches such as (sparse) resultants, Groebner bases, and methods based on eigenvalue computations or on *rational univariate representation*.

Homotopy methods numerically track the continuous path of the known complex solutions of some trivial and appropriate polynomial system during a continuous deformation into the input system. Such methods, although very robust, lack the certification of their output in general. We recommend [26] for a more comprehensive overview.

Subdivision methods describe a further class of common tools. Algorithm of that kind profit from their efficiency and plainness. Most Implementations are using one of the numerous software packages for efficient interval arithmetic [2, 18, 20], such as IntBis, ALIAS, IntLab or MPFI. Alternative variants, using the Bernstein basis and convexity properties of their coefficients, have been addressed [19]. However, all these approaches lack to certify their results – in general, an approach stops when a certain subdivision depth is reached or each region contains a simple root, which can, for instance, be certified by the interval Newton test [26, Sec. 6.1]. So far, in case of multiple roots, all proposed methods have to go below a certain a-priori worst case root separation bound in order to certify that a region is isolating. As a result of the bad quality of these bounds in the average case, subdivision methods turn out to be impractical for an exact and complete approach.

*In case of a univariate polynomial f. For the general case, we consider $|\mathrm{sres}_{k_j}(r_j,r_j')|$, where r_j is the elimination polynomial of $F=0$ with respect to z_j and $k_j := \deg\gcd(r_j,r_j')$.

Also specializations of the general problem have been extensively studied. The probably most prominent one is univariate polynomial solving. In particular, this is an integral building block in elimination methods, where the (univariate) elimination polynomial has to be considered, and lifting the solutions to higher dimensions usually leads to further univariate systems.

Certified algorithms for real root isolation are mainly subdivision solvers based on Descartes' Rule of Signs or Sturm sequences, see [3, 10, 29] for modern approaches. For the complex case we refer to [16, 21, 23, 24, 25].

Other special cases of polynomial systems appear in real algebraic geometry. Recent implementations for computing the topology of algebraic curves either make use of elimination methods [11, 13] or subdivision [1, 7]. Already this low dimensional application shows the mentioned drawbacks of the distinct approaches: subdivision fails to give a certified result in degenerate cases whereas elimination methods suffer from costly symbolic computations.

Outline. We sketch our algorithm in Section 2. Therein, we refer to the Sections 3-6 for the details of its submethods. Section 7 finally concludes our results.

2 Our approach

We fix the following notation throughout the paper: our input system is given by polynomials $f_i \in \mathbb{Z}[z_1,\ldots,z_N]$ of total degrees d_i, $i=1,\ldots,N$, and $D=\prod_{i=1}^{N} d_i$ defines their product. Then these polynomials induce a function $F:\mathbb{C}^N\to\mathbb{C}^N$ that maps a point $p\in\mathbb{C}^N$ to $(f_1(p),\ldots,f_N(p))$. By assumption, F has only finitely many isolated solutions (or roots), let $\Gamma=V(F)$ denote them. For $j=1,\ldots,N$, let $\pi_j:\mathbb{C}^N\to\mathbb{C},(z_1,\ldots,z_N)\mapsto z_j$ denote the canonical projection map with respect to z_j. Likewise, $\Gamma_j:=\pi_j(\Gamma)$ is the set of z_j-coordinates of solutions of F.

Given F, our algorithm computes disjoint *isolating clusters* $C_1,\ldots,C_s\subset\mathbb{C}^N$, that is, each cluster C_i contains precisely one root of F and $\Gamma\subset\bigcup_{i=1}^{s} C_i$. In general, we will use the term *cluster* for a connected subset of $\mathbb{C}^N$.

`Transformation phase` (Section 3.1): As a first step, the algorithm ensures several genericity conditions[†] which further steps rely on. Matrices $M_1,\ldots,M_N$ with $M_j\in\mathbb{Z}[z_j]^{\ell_j\times\ell_j}$ (for some ℓ_j) are computed whose determinants $r_j\in\mathbb{Z}[z_j]$ satisfy the following properties: r_j describes the projected solutions of F with respect to the coordinate z_j, in short $V(r_j)=\Gamma_j$, and multiplicities are preserved under projection. Moreover, each r_j must be of degree D; this ensures that F has no solution at infinity, that is, all solutions of its homogenization are contained in $\mathbb{C}^N$, considered as an affine chart of $\mathbb{P}^N$. All these properties are tried to be ensured only using modular arithmetic (i.e., without computing the r_j's exactly). If this fails, the algorithm starts over with a transformation of F by a linear projective change of coordinates.

The actual isolation algorithm depends on the following three subroutines.

`Guess_`k_j (Section 3.2): For $j\in\{1,\ldots,N\}$, it computes an upper bound k'_j for $k_j:=\deg\gcd(r_j,r'_j)$, which also yields a lower bound m'_j on the cardinality $m_j=D-k_j$ of Γ_j. This is done by computing the index of the first non-vanishing principal subresultant coefficient of r_j and r'_j in a modular domain $\mathbb{Z}_p$, where the prime p is newly chosen in each call.

`Subdivide` (Section 4): Returns a set of disjoint clusters that cover all roots of F, and each contains at least one root (we call clusters known to contain at least one root *zero-clusters*). This is done by subdivision in $\mathbb{C}^N\cong\mathbb{R}^{2N}$ to exclude regions without roots, combined with a criterion to identify zero-clusters. Repeated calls of `Subdivide` shrink the zero-clusters, and if a zero-cluster is not isolating, it will split into several parts after sufficiently many steps.

[†] for $N=1$, the phase can be skipped

`Mult_of_clusters` (Section 5): For the zero-clusters C returned by `Subdivide`, the number $\mu(C)$ of roots of F inside each cluster is computed, counted with multiplicity (notice that $\mu(C) \geq 2$ does not imply that C is non-isolating, since multiple roots can occur). The method slightly perturbs F into $\tilde{F}$ such that roots remain in their zero-clusters, and such that a root of multiplicity μ turns into μ simple roots nearby. The number of roots inside a zero-cluster is than counted by further subdividing the cluster.

`Main routine:` Our main root isolation routine passes through two phases. First, the *synchronization phase* (Section 6.1) calls `Guess_k`$_j$ repeatedly for each projection direction to obtain a lower bound m'_j on m_j, Though it is very likely to coincide with m_j, m'_j might improve in further calls. In parallel, it calls `Subdivide` repeatedly to get smaller and smaller zero-clusters $C_1, \ldots, C_{s'}$. For each direction it examines $\pi_j(\bigcup C_i)$, which decomposes into connected components, called *projected clusters*. Once the number of projected clusters under π_j coincides with m'_j for each j, `Mult_of_clusters` is called for $C_1, \ldots, C_s$ (this also yields multiplicities of projected clusters), and the algorithm switches to the *certification phase*, described next.

The clusters $C_1, \ldots, C_{s'}$ are isolating under the condition that $m'_j = m_j$, or equivalently $k'_j = k_j$, for each j. The *certification phase* (Section 6.2) tries to verify this. It computes the values LB_j, that is, the product of the primes used in `Guess_k`$_j$ so far, and UB_j which is determined by the diameters and multiplicities of projected clusters, and the distances between them. If $k'_j > k_j$, LB_j (UB_j) is a lower (upper) bound for $|\mathrm{sres}_{k_j}(r_j, r'_j)|$. The algorithm again calls `Guess_k`$_j$ repeatedly, which makes LB_j arbitrary large, and in parallel, it calls `Subdivide` which lets UB_j converge to zero. Either, $LB_j > UB_j$ at some point which algorithmically proves by contradiction that $k'_j = k_j$. Or, a call of `Guess_k`$_j$ improves the upper bound on k_j, or a call of `Subdivide` makes a projected cluster split into two parts. Both cases disprove $k'_j = k_j$ and thus, clusters were not isolating yet; the algorithm then switches back to the synchronization phase.

3 Symbolic tools

3.1 *Transformation phase*: Multivariate Resultant [‡]

Crucial for our method is the knowledge of univariate polynomials $r_1, \ldots, r_N$ such that the roots of r_j are precisely the z_j-coordinates of points in Γ. Each r_j is represented as the determinant of a matrix M_j in the coefficients of $f_1, \ldots, f_N$. Also, each r_j should have degree D, which ensures that all D solutions of the homogenized system over $\mathbb{P}^N$ lie in the considered affine chart.

It is possible that such r_j's (more precisely, the M_j's) only exist after a projective coordinate transformation, and in degenerate instances our method might even fail completely to compute such M_j's. We closely follow the ideas described in [9, § 3.5], using multivariate resultants and the "hidden variable" approach. Here, we just mention the main facts from the theory; see [9, 12] for further explanations. We first introduce the (affine) multivariate resultant:

Definition 1. *For a system of n polynomials $(f_1, \ldots, f_n)$ in $n-1$ variables, $\mathrm{res}_{(f_1,\ldots,f_n)}$ is an irreducible polynomial in the coefficients of the f_i's. The resultant vanishes if and only if the homogenized system has a solution over $\mathbb{P}^n$.*

To apply the multivariate resultant in our problem with N equations in N variables, we consider one variable z_j as a parameter, that means, the system has coefficients in $\mathbb{Z}[z_j]$. Then, $\mathrm{res}_j := \mathrm{res}^{z_j}_{(f_1,\ldots,f_N)}$, $j = 1, \ldots, N$, defines a polynomial in z_j. From the definition, it follows that its roots are precisely the z_j-coordinates where $F = 0$ has a solution.

[‡] for $N = 1$, the whole section can be skipped

Theorem 2. *Let* $\deg(\mathrm{res}_j) = D$ *for all* $j = 1,\ldots,N$. *Then* F *has precisely* D *roots in* $\mathbb{C}^N$, *counted with multiplicity*[§]. *For each* $(z_1,\ldots,z_N) \in \Gamma$, z_j *is a root of* res_j *and the multiplicity of* z_j *is the sum of the multiplicities of the points in its fiber.*

Proof. The degree of each res_j is bounded by D (cf. [12]). If any solutions is not finite, at least one res_j must have a "root at infinity", thus its degree drops by at least one, which shows (1). The second claim follows directly from the definition of the multivariate resultant and the last from the previous, noting that the resultant is continuous in the coefficients of the system, and that a slight perturbation of the system yields D simple solutions with distinct z_j-coordinates. □

In our algorithm, we exploit that res_j can be represented, at least generically and up to a constant, as the determinant of a coefficient matrix of $(f_1,\ldots,f_N)$ by a theorem due to Macaulay [17].

Theorem 3. *There exist matrices* M_j *and* M'_j *whose entries are polynomials in the coefficients of the system* $(f_1,\ldots,f_N)$ *and in* z_j, *such that* $\mathrm{res}_j = \frac{\det M_j}{\det M'_j}$. M'_j *does not contain* z_j.

For the definition of M_j and M'_j, see [9]. Theorem 3 implies that if for our concrete system $\det M'_j \in \mathbb{C}$ does not vanish, then $r_j := \det M_j$ equals res_j up to a constant. At several places in our algorithm we need to upper bound the coefficients of r_j. Using the Hadamard bound on M_j, one obtains the following estimation:

Lemma 4. *The bitsize of the coefficients of each* r_j *is at most* $\sigma := n(\tau + \log n) + \log(dn+1)$, *where* τ *is the maximal bitsize of any coefficient of* $(f_1,\ldots,f_N)$, $n = \binom{\sum_{i=0}^{N} d_i}{N-1}$ *and* $d = \max d_i$.

Proof. The dimension of M_j equals $\binom{\sum_{i\neq j} d_i}{N-1} < n$ (cf. [9]). Moreover, each entry of M_j is a univariate polynomial whose coefficients have bitsize at most τ, and whose degree is bounded by d. Using [3, Prop. 8.12], the determinant polynomial r_j has thus a bitsize bounded by σ. □

We next describe the transformation phase of the algorithm. All computations are performed modulo a prime number p. Let $\bar{\cdot}$ denote the operation that maps integers to their modular image in $\mathbb{Z}_p$. This map extends to integer polynomials and integer matrices in the obvious way.

Choose a random prime p and compute over $\mathbb{Z}_p$, for each $j = 1,\ldots,N$, $\det \overline{M'_j} \in \mathbb{Z}_p$ and $\det \overline{M_j} \in \mathbb{Z}_p[z_j]$. If any determinant $\det M'_j$ vanishes, or for any j, $\deg(\det M_j) \neq D$, transform $(f_1,\ldots,f_N)$ via a random linear projective change of coordinates and start over with the transformed system.

We remark that the transformation phase might loop forever, in case that $\det M'_j$ vanishes for the input system, and for all its linear projective transformations.

For special cases of polynomial systems, res_j is explicitly known as determinant of a matrix, not just as a quotient. The most prominent case is $N = 2$, where the *Sylvester matrix* can be used; Theorem 3 and consideration of M'_j is not needed. Special cases with $N > 2$ are discussed in [28].

A perhaps unpleasant feature of the transformation is that also infinite solutions of the original system are computed. It is possible to rule out them in a post-processing step after the algorithm. For that, we consider a finite region that contains all finite roots of the original coordinate system. For instance, it is possible to consider the box $[-2^\sigma, 2^\sigma]^{2N}$ with σ from Lemma 4.

[§]The multiplicity of a root $\xi \in \mathbb{C}^N$ of F is defined as the dimension of the localization of $\mathbb{C}[z_1,\ldots,z_N]/(f_1,\ldots,f_N)$ at ξ considered as a $\mathbb{C}$-vector space (cf.[3, p.148])

3.2 *Guess_k$_j$*: Modular computation

We turn to the method `Guess_k`$_j$ that computes an upper bound k'_j on $k_j = \deg\gcd(r_j, r'_j)$. Since $\deg(r_j) = D$ (guaranteed by the transformation phase), $D - k'_j$ constitutes a lower bound m'_j on m_j, the number of distinct roots of r_j. To compute k'_j, we exploit the relation (e.g. [3, Prop. 4.25])

$$k_j := \deg\gcd(r_j, r'_j) = \min\{i \geq 0 \mid \mathrm{sres}_i(r_j, r'_j) \neq 0\},$$

where $\mathrm{sres}_i(f,g)$ denotes the i-th principal subresultant coefficient of f and g.

By definition, $\mathrm{sres}_i(r_j, r'_j)$ can again be expressed as determinant in the coefficients of r_j. Therefore, its computation is possible in a modular domain $\mathbb{Z}_p$ for a prime p. Defining λ_j as the leading coefficient of r_j, we obtain the following.

Proposition 5. *Let $j \in \{1,\ldots,N\}$, and p a prime, that does not divide λ_j or $D \cdot \lambda_j$, the leading coefficient of r'_j. Then, for $i = 0,\ldots,D$ we get* $\mathrm{sres}_i(\overline{r_j}, \overline{r'_j}) = \overline{\mathrm{sres}_i(r_j, r'_j)}$.

Lemma 6. *Let $p_1,\ldots,p_s$ be distinct prime numbers that do not divide $D\lambda_j$, and*

$$k'_j := \min_{\ell=1,\ldots,s} \min\{i \geq 0 \mid \mathrm{sres}_i(\overline{r_j}, \overline{r'_j}) \neq 0\},$$

where $\bar{\cdot}$ is the modular operation with respect to p_ℓ. Then, $k'_j \geq k_j$, and thus r_j has at least $m'_j := D - k'_j$ distinct complex roots. Moreover, for each $i \in \{0,\ldots,k'_j - 1\}$,

$$\mathrm{sres}_i(r_j, r'_j) = 0 \vee |\mathrm{sres}_i(r_j, r'_j)| \geq \prod_{\ell=1}^{s} p_\ell.$$

The second part of Lemma 6 constitutes a lower bound for the size of non-vanishing principal subresultants coefficients. We will exploit this lower bound in the certification phase (Section 6.2). We explain our method *Guess_k$_j$* next. For any $j \in \{1,\ldots,N\}$ it outputs a pair consisting of an upper bound $k_j^{(p)}$ for k_j, and the prime p that has been used to obtain this bound.

Choose a prime p not considered so far, compute $\overline{r_j} = \det \overline{M_j}$ and $\overline{r'_j}$, with respect to p, until $\deg \overline{r_j} = D$ and $\deg \overline{r'_j} = D - 1$. Compute $k_j^{(p)} = \min\{i \geq 0 \mid \mathrm{sres}_i(\overline{r_j}, \overline{r'_j}) \neq 0\} = \deg\gcd(\overline{r_j}, \overline{r'_j})$, and return the pair $(k_j^{(p)}, p)$.

Note that `Guess_k`$_j$ performs all computations in the domain $\mathbb{Z}_p$, no exact evaluation of r_j is necessary. The price we pay is that we have to cope with the uncertainty whether $k'_j = k_j$ holds or not. This guess must be checked at the end of the overall algorithm. However, we claim that a wrong guess is very unlikely since $k_j = k'_j$ will hold as soon as a prime p is chosen that does not divide $\mathrm{sres}_{k_j}(r_j, r'_j)$.

4 *Subdivide*: Subdivision

We apply a subdivision scheme on $\mathbb{C}^N \cong \mathbb{R}^{2N}$ to identify clusters containing roots of F. Writing each $z_j := x_j + i \cdot y_j$ and $f_j = g_j + i \cdot h_j$, F can be interpreted as a function $F : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$ that maps a point $p = (x_1, y_1, \ldots, x_N, y_N) \in \mathbb{R}^{2N}$ to $(g_1(p), h_1(p), \ldots, g_N(p), h_N(p))$. For a box $A = [a_1, b_1] \times [c_1, d_1] \ldots \times [a_N, b_N] \times [c_N, d_N]$, let $\Box F(A)$ be the result of evaluating F at A in interval arithmetic,

e.g., by the use of the recursive Horner scheme or centered box evaluation (also denoted as modified affine arithmetic [14]). By the properties of interval arithmetic [20], we get $\mathrm{Im}(\mathrm{F}|_{\mathrm{A}}) \subset \square F(A)$. We call a box A *hot*, if $\square F(A)$ contains the origin. Clearly, non-hot boxes do not contain any root of F.

We start with an initial box containing all roots.[¶] In each iteration, every hot box B is subdivided into 2^{2N} even parts, which replace the old box B. All new boxes are tested to be hot, non-hot boxes are removed. In each state the hot boxes can be grouped into maximal connected regions, called *hot clusters*. In each iteration, a hot cluster either splits into smaller hot clusters, dies (i.e., vanishes completely), or persists, that means, it remains connected after the subdivision step. However, it is not clear whether a hot clusters indeed contains a root. We derive a method to ensure the presence of at least one root inside a hot cluster next.

Theorem 7. *Let $D \subset \mathbb{C}^N$ be an open, connected subset. If there exists a point $p \in D$ such that $|F(p)| < m_{\partial D} := \min_{\gamma \in \partial D} |F(\gamma)|$, then F has a root in D.*

Proof. Consider the parameterized function $F_t = F - tF(p)$ for $t \in [0,1]$. Then the roots of F_t continuously depend on t and F_1 has a root, namely p, within D. When passing from F_1 to $F = F_0$ this root continuously transforms into a root p' of F. If p' is outside D, then there must exist a $t_0 \in [0,1]$ such that F_{t_0} has a root p^* on ∂D. But, $0 = |F_{t_0}(p^*)| = |F(p^*) - t_0 F(p)| \geq ||F(p^*)| - |t_0 F(p)|| \geq m_{\partial D} - |F(p)| > 0$, thus p' is also located in D. □

Definition 8. *Let C be a hot cluster consisting of hot boxes $B_1, \ldots, B_s$. We define $\Delta(C)$ as the union of all boxes $\tilde{B}_1, \ldots, \tilde{B}_r$ that are adjacent to C. We define*

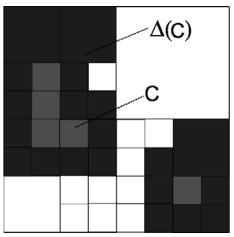
$$\varepsilon(C) := \min_{j=1,\ldots,s} \left|F(\rho_j))\right|$$

where ρ_j denotes the midpoint of B_j. Furthermore

$$\delta(C) := \min_{l=1,\ldots,r} \left|\square F(\tilde{B}_l)\right| > 0,$$

if $\partial C \subset \partial\Delta(C)$ and $\delta(C) := 0$, otherwise.

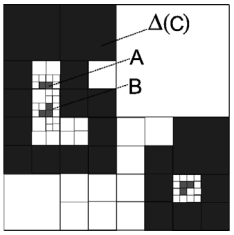
Notice that $\partial C \subset \partial\Delta(C)$ exactly holds iff none of the boxes B_j has a common point with the boundary of the initial box. In this situation we get $\min_{\gamma \in \partial C} |F(\gamma)| \geq \delta(C) > 0$. Hence, by Theorem 7, we can conclude that a cluster C contains a root of F if $\varepsilon(C) < \delta(C)$. For a sequence of clusters C_k that approximates a root ξ it is clear that $\varepsilon(C_k) \to 0$ as well as $\delta(C_k) \to 0$ while $k \to \infty$. This does not imply that we reach a state such that $\varepsilon(C_k) < \delta(C_k)$. But for some fixed k' there is a $k'' \geq k'$ such that $\varepsilon(C_{k''}) < \delta(C_{k'})$



Corollary 9. *Let C' and C be two hot clusters and C' be a descendant cluster of C, that is, $C' \subset C$. If $\varepsilon(C') < \delta(C)$ then C contains a root of f.*

To illustrate how Corollary 9 is applied, consider the picture on the right. Assume that the cluster C splits during the subdivision, and yields two hot sub-clusters A, B. If $\varepsilon(A) < \delta(C)$, then C must contain a root. However, it does not imply that A also contains a root, it only follows that $A \cup B$ contains a root.

[¶] e.g., one can use the box $[-2^\sigma, 2^\sigma]^{2N}$ with σ as in Lemma 4



We introduce a data structure that will help to identify clusters that are certified to contain a root of F. For that, we maintain a tree $\mathscr{T}$, called *cluster tree*, where each node v_C in $\mathscr{T}$ corresponds to a hot cluster C and a node $v_{C'}$ is a child of v_C if $C' \subset C$ is a hot cluster of the next iteration step. The root of $\mathscr{T}$ corresponds to the initial box. Moreover, each node in the tree maintains a δ and an ε-value that are initialized according to Definition 8. While δ does not change, new descendent nodes can give rise to better ε-values which are propagated to all their ancestors. Once $\varepsilon < \delta$ for a node, the corresponding cluster is known to contain a root by Theorem 7. We tag such a node and all its ancestors with a *zero* flag. We call these nodes *zero-nodes*, and the corresponding clusters *zero-clusters*. It may also happen that all descendent nodes of a node v die out. In this case the subtree which is rooted at v is completely removed from $\mathscr{T}$. Hence, the leaves of $\mathscr{T}$ are in one-to-one correspondents to the hot clusters in the current subdivision state.

Proposition 10. *A hot cluster that contains a root of F will eventually be a zero-cluster in the cluster tree. A hot cluster that does not contain a root of F will eventually be removed from the cluster tree.*

Moreover, we maintain a subtree $\mathscr{T}'$. We start with the root node of $\mathscr{T}$ and grow $\mathscr{T}'$ as follows. Add all children of a leaf of $\mathscr{T}'$ as soon as all of them are zero-nodes. By construction, the leaves of $\mathscr{T}'$ correspond to a set of disjoint zero-clusters, which covers all roots of F. We call this set the current *minimal zero-cluster overlay* (*MZCO*).

Proposition 11. *The* MZCO *constitutes a set of isolating clusters for F after sufficiently many subdivisions.*

A call of the method `Subdivide` triggers another subdivision step, updates $\mathscr{T}$, $\mathscr{T}'$ and finally returns the current *minimal zero-cluster overlay*.

---

5 *Mult_of_clusters*: Perturbation

The multiplicity of a root $\xi \in \mathbb{C}^N$ of F is defined as the dimension of the localization of the ring $\mathbb{C}[z_1,\ldots,z_N]/(f_1,\ldots,f_N)$ at ξ, considered as a $\mathbb{C}-$vector space (cf.[3, p.148]). A more intuitive description states that ξ is a root of multiplicity μ if there exists a neighborhood $U(\xi) \subset \mathbb{C}^N$ such that almost any sufficiently small perturbation of F has exactly μ distinct, simple roots in $U(\xi)$.

We next discuss `Mult_of_clusters` that computes the sum of the multiplicities of all roots inside each $C \in \mathscr{C}$, where $\mathscr{C}$ is a MZCO as returned by the method `Subdivide`. We first choose a *perturbation vector* $\upsilon \in \mathbb{Q}^{2N}$ such that $\forall C \in \mathscr{C} : |\upsilon| < \delta(C)$. The following Lemma ensures that roots can not leave their cluster when perturbing by υ.

Lemma 12. *Let C be some hot cluster and $\upsilon \in \mathbb{R}^{2N}$ with $0 < |\upsilon| < \delta(C)$, then the number of roots of $F + \upsilon$ equals the number of roots of F in C.*

Proof. Consider the parameterized function $F_t = F + t\upsilon$ and proceed as in Theorem 7. □

We also ensure that υ is chosen such that all roots of $\tilde{F}$ are simple. One possibility is to check whether $\mathrm{res}(\tilde{r}_1, \tilde{r}_1') \neq 0$ over some $\mathbb{Z}_p$, p a random prime.‖ In case of a failure we simply choose another υ and perform the modular computation with some other prime until we succeed.

‖For the definition and the computation of the resultant $\tilde{r}_1$ with respect to $\tilde{F}$ we refer to Section 3.

Once υ is chosen, we know that all roots of $\tilde{F}$ are simple. In particular, we know that $\tilde{F}$ has precisely D simple roots. This is guaranteed by the fact that F, and therefore $\tilde{F}$, has only finite roots, see also Section 3.1. Hence, we can apply a variant of our subdivision scheme to $\tilde{F}$ as follows:

For a given MZCO $\mathscr{C} := \{C_1, \ldots, C_s\}$ for F the method `Mult_of_clusters` computes $\tilde{F}$ for some proper perturbation vector $\upsilon \in \mathbb{Q}^{2N}$. Now the subdivision method as discussed in Section 4 is applied to $\tilde{F}$ with initial clusters $C_1, \ldots, C_s$ inducing a forest of cluster trees. The subdivision is continued until the MZCO defined by the forest precisely contains D clusters. The number of zero-clusters in the *MZCO* with root C_i then gives the number $\mu(C_i)$ of roots of F in the cluster C_i, counted with multiplicities.

Note that the method `Mult_of_clusters` can be modified such that it uses a separate perturbation vector υ_i for each cluster $C_i \in \mathscr{C}$. That is, each υ_i fulfills $|\upsilon_i| < \delta(C_i)$ and all roots of $F + v_i$ within C_i are simple.

Improved Perturbation Vectors: So far, we sketched a method to compute a global, randomly chosen perturbation vector υ. Although it is already guaranteed that $\tilde{F}$ has only simple roots, an unfortunate choice of υ leads to a bad root separation, and hence to a high subdivision depth for $\tilde{F}$. To overcome this problem, we next propose an alternative approach. For each $C_i \in \mathscr{C}$, it computes a perturbation vector υ_i independently. It makes use of a criterion to guarantee that an axis-aligned box contains at most one root, which is simple. We introduce some notation first. It is well known (see [3, Prop. 4.94] for a proof) that the Jacobian $J_F := (\partial f_j/\partial z_k)_{1\le j,k\le N}$ of F at ξ has full rank if ξ is a simple root of F. For its real counter part

$$D_F := \begin{pmatrix} \frac{\partial g_j}{\partial x_k} & \frac{\partial g_j}{\partial y_k} \\ \frac{\partial h_j}{\partial x_k} & \frac{\partial h_j}{\partial y_k} \end{pmatrix}_{1\le j,k\le N}$$

it holds that $\det(D_F) = \det(J_F)^2$ [15, p.27], thus it follows:

Lemma 13. *F has a simple root at $\xi \in \mathbb{C}^N$ iff $F(\xi) = 0$ and $\det(D_F)(\xi) = (\det(J_F)(\xi))^2 \neq 0$.*

For $n > 1$, Rolle's theorem is not directly applicable to functions $\varphi : \mathbb{R}^n \mapsto \mathbb{R}^n$, thus F might have two roots without a root of $\det(D_F)$ in between. But we can use another criterion that exploits the behavior of interval arithmetic. We consider the matrix D_F and denote its row vectors by

$$G_j := \left(\frac{\partial g_j}{\partial x_i}, \frac{\partial g_j}{\partial y_i}\right)_{1\le j\le N}, \quad H_j := \left(\frac{\partial h_j}{\partial x_i}, \frac{\partial h_j}{\partial y_i}\right)_{1\le j\le N}$$

Then for any $\Phi := (p_1, \ldots, p_N, q_1, \ldots, q_N) \in \left(\mathbb{R}^{2N}\right)^{2N}$, we define $M(\Phi)$ as the $2N \times 2N-$matrix whose rows are the vectors $G_j(p_j)$ and $H_j(q_j)$, $j = 1, \ldots, N$.

Lemma 14. *Let $B \subset \mathbb{R}^{2N}$ be an axis-aligned box and $D : \left(\mathbb{R}^{2N}\right)^{2N} \to \mathbb{R}^{2N}$ defined by $D(\Phi) := \det(M(\Phi))$. Then*

1. *If $0 \notin \square D(B^{2N})$, then B contains at most one root of F, which is simple.*
2. *If B contains exactly one root of F (counted with multiplicity) and B is sufficiently small, then $0 \notin \square D(B^{2N})$.*

Proof. We consider the case $N = 1$ where $F : \mathbb{C} \to \mathbb{C}$ is a univariate polynomial. Then the general case follows in a complete analog manner. F can be written as $F(x+iy) = f(x,y) + ig(x,y)$ with

polynomials $f,g \in \mathbb{Z}[x,y]$. If $B \subset \mathbb{R}^2$ contains a multiple root a of F, then $D(a,a) = D_F(a) = 0$, thus we can restrict to the case where B contains two distinct roots a,b. We denote $[a,b]$ the line segment connecting a and b. From Rolle's theorem in several real variables, applied to the polynomials f and g, it follows the existence of points $p,q \in [a,b]$ with

$$\begin{pmatrix} f_x(p) \\ f_y(p) \end{pmatrix} \cdot (a-b) = \begin{pmatrix} g_x(q) \\ g_y(q) \end{pmatrix} \cdot (a-b) = 0.$$

Thus, $(f_x(p), f_y(p))^t$ and $(g_x(p), g_y(p))^t$ are perpendicular to $a-b$, which is only possible if they are linearly dependent. As a consequence we must get

$$\det \begin{pmatrix} f_x(p) & f_y(p) \\ g_x(q) & g_y(q) \end{pmatrix} = 0$$

Hence, it follows that $0 \in \Box D(B^2)$. In contrast, if B has only one simple root $a \in B$ then $\det(D_F|_p) \neq 0$. D is continuous, thus if we choose B small enough, this guarantees that $0 \notin \Box D(B^2)$. □

Note the subtle difference of Lemma 14 compared with the criterion $0 \notin \Box \det D_F(B)$ which does not guarantee the presence of at most one root in B; it is needed that the values in each row of the matrix are chosen independently of each other.

Using Lemma 14 we compute for each cluster $C \in \mathscr{C}$ a corresponding perturbation vector v_C. Compared to the modular approach, this leads to a better separation for the roots of $\tilde{F} := F + v_C$ within C. For a box B we define its expanded box B^+ as the box with the same center as B, and thrice as large side length. We call a box *simple*, if $0 \notin \Box D((B^+)^{2N})$, and *non-simple* otherwise.

Copy all boxes of C that are non-simple into a new subdivision queue. Start subdividing, and keep only non-simple boxes in the queue. In addition, evaluate $\Box F(B)$ for each box in the queue, let $U \subset \mathbb{C}^N$ be the union of these interval vectors. Stop the subdivision as soon as $\mathbb{R}^{2N} \setminus U$ contains a vector v with $|v| \in [0, \delta(C)]$. Set $v_C := v$ and return.

Why does the above algorithm terminate? Notice that $\det D_F = 0$ describes a hypersurface $V(det(D_F))$ in $\mathbb{C}^N$, thus the function values of F on $V(det(D_F))$ also describe a hypersurface in $\mathbb{C}^N$. It follows that after finitely many subdivision steps, the complement of the union U of interval vectors $\Box F(B)$ must contain a vector v with $|v| \in [0, \delta(C)]$.

Lemma 15. *For a cluster C, let v_C be chosen as by the algorithm above, then $\tilde{F} := F + v_C$ has only simple roots within C and all these roots are separated by at least s, where s denotes the minimal side length of all boxes considered during the subdivision.*

Proof. Note first that $D\tilde{F} = DF$, thus (non-)simple boxes remain (non-)simple when switching to $\tilde{F}$. From the choice of v_C, it follows that each $\xi \in C$ with $F(\xi) = v_C$ is contained in one of the simple boxes. Hence, all roots of $\tilde{F}$ within C are simple as well. Now consider two roots a,b of $\tilde{F}$, then both of them are contained in simple boxes B_1 and B_2 respectively. As none of the boxes B_1 or B_2 contains two roots we get that $B_1 \neq B_2$ and B_1, B_2 are not adjacent. □

We remark another application of Lemma 14, although it is not directly needed for our application. Observe that the method *Mult_of_clusters* relies on the fact that all clusters are considered simultaneously, because it stops as soon as the total number of certified clusters equals D.

With a slight modification it is possible to describe a local version of *Mult_of_clusters* that computes the number of roots, counted with multiplicity, within a given cluster C. It computes a ΔC as

before, and subdivides C with respect to the evaluation function $F_{\Delta C}$. Also, a cluster tree is maintained for C to identify zero-clusters, as before. For each zero-cluster $C_0 \subset C$ in the subdivision, a box B of minimal size is computed that completely contains this cluster. If $0 \notin \square D(B^{2N})$, the zero-cluster contains precisely one simple root, we call it *simple cluster*. If a cluster is simple, all its subclusters are simple as well. If no such box B can be computed, further subdivision is necessary. The algorithm stops if each element of the current MCZO is simple.

6 The main routine

6.1 *Synchronization phase*: Projection

In Section 3.2, we explained a method `Guess_k`$_j$ to obtain a lower bound m'_j on m_j, the number of distinct roots of r_j. In Section 4, we presented a method `Subdivide` to compute a set of disjoint clusters in $\mathbb{C}^N$, containing all roots of F, and each set contains at least one solution (we called such clusters *zero-clusters*). In the synchronization phase, both subroutines are combined to make their outcome coherent. For that, we introduce the notion of a *projected cluster*. Recall that, for all $j \in \{1,\ldots,N\}$, the roots of r_j coincide with Γ_j. Therefore, for a zero-cluster $C \subset \mathbb{C}^N$, $\pi_j(C) \subset \mathbb{C}$ is a connected region that contains at least one root of r_j. When projecting all clusters returned by `Subdivide`, some clusters might overlap in the projection. We therefore define

Definition 16. *Let $\mathscr{C} = \{C_1,\ldots,C_s\}$ be a set of zero-clusters returned by* `Subdivide`*. The* projected clusters *under π_j are the maximal connected components of $\bigcup_{C\in\mathscr{C}} \pi_j(C)$. The* multiplicity *$\mu(R)$ of a projected cluster R is defined as*

$$\mu(R) = \sum_{C\in\mathscr{C},\pi_j(C)\subset R} \mu(C),$$

where $\mu(C)$ is the multiplicity of the cluster C.

By the properties of r_j (Theorem 2), it follows

Lemma 17. *Let R be a projected cluster under π_j. Then R contains exactly $\mu(R)$ roots of r_j, counted with multiplicity. In particular, the set of projected clusters covers all roots of r_j and each projected cluster contains at least one root.*

During the subdivision, the clusters in $\mathbb{C}^N$ become arbitrary small, and the same holds for their projections. Thus, after sufficiently many calls of `Subdivide`, the returned clusters will induce exactly m_j projected clusters under π_j.

Lemma 18. *Let $\mathscr{C} = \{C_1,\ldots,C_s\}$ be a set of zero clusters returned by* `Subdivide`*, such that, for each j, there are m_j projected clusters under π_j. Then each cluster C_i contains exactly one root.*

Proof. If a cluster C_i contains two solutions, they differ in at least one variable z_j, and so, r_j has more than m_j distinct solutions, a contradiction. □

Our algorithm, however, does not apply Lemma 18 directly, since computing the m_j's is too costly. Instead, the synchronization method both computes a lower bound m'_j on m_j, and performs subdivision in $\mathbb{C}^N$, until there are precisely m'_j projected clusters with respect to z_j for each $j = 1,\ldots,N$. In the subsequent certification phase, it will be verified (or falsified) that $m'_j = m_j$ holds.

Here is the detailed description of the synchronization phase. The algorithm stores for each j a *current guess* m'_j, initially set to 0, and a number LB_j, initially set to one (the latter will be used in the certification). Also, it initially calls `Subdivide` and stores the returned set into $\mathscr{C}$.

Repeat the following steps: For each $j = 1, \ldots, N$, compute the projected clusters for $\mathscr{C}$ under π_j, let c_j denote their number. If $m'_j = c_j$ for all j, call `Mult_of_clusters` for each $C \in \mathscr{C}$, compute the multiplicity of each projected cluster according to Definition 16, and pass to the certification phase. Otherwise, let j be such that $m'_j \neq c_j$. If $m'_j < c_j$, call `Guess_k`$_j$, let $(k_j^{(p)}, p)$ be the result. Set m'_j to $\max\{m'_j, D - k_j^{(p)}\}$, set LB_j to $LB_j \cdot p$, and proceed with the next iteration. If $m'_j > c_j$, call `Subdivide`, set $\mathscr{C}$ to its output, and proceed with the next iteration.

Both m'_j and c_j are lower bounds for m_j, and after sufficiently many calls of `Guess_k`$_j$ or `Subdivide`, respectively, both will be set to m_j. However, the algorithm might jump to the certification phase earlier, when $c_j = m'_j \neq m_j$. As we will see, the certification phase will then falsify $m'_j = m_j$, and increase at least one of the values $c_1, \ldots, c_N, m'_1, \ldots, m'_N$.

6.2 *Certification phase:* Separation bounds

As Lemma 18 states, the clusters computed by the synchronization phase contain precisely one root of F under the condition that $m'_j = m_j$, or equivalently $k'_j = k_j$, for each j. For the sake of simpler notation in this subsection, we will fix one projection variable z_j, and set $k := k_j$, $k' := k_j$, $m := m_j$, $m' := m'_j$, and $r := r_j$. Note that $D := \deg(r)$. Also, we will denote the projected clusters for r by $R_1, \ldots, R_{m'}$. Our goal is to prove (or disprove) $k' = k$.

Our certification scheme uses two dynamically changing values $LB := LB_j$ and $UB := UB_j$ with the following properties:

- If $k' > k$, it holds that $LB \leq |\mathrm{sres}_k(r, r')| \leq UB$.
- If $k = k'$, further calls of `Guess_`k_j improve LB. More precisely, a sequence of calls of `Guess_`k_j leads to values LB that diverge to $+\infty$.
- If $k = k'$, further calls of `Subdivide` improve UB. More precisely, a sequence of calls of `Subdivide` leads to values UB that converge to 0.

The idea for the certification is to call `Guess_`k_j and `Subdivide` simultaneously (or alternating), until $LB > UB$ which proves that $k' = k$. If a call of `Guess_`k_j decreases k', or if any call of `Subdivide` leads to a split of some cluster R_i, the guess is falsified. In this case, the algorithm has to return to the synchronization phase to produce a new guess for k.

How do we obtain such bounds LB and UB? For LB, the answer is simple. Recall that Lemma 6 provides a lower bound for each non-vanishing $|\mathrm{sres}_l(r, r')|$ with $l < k' = k'_j$, namely the product of the primes considered so far by `Guess_`k_j. The algorithm keeps track of this product by the variable LB_j, compare the description of the synchronization phase. Clearly, each call of `Guess_`k_j makes LB larger, and the product diverges to $+\infty$.

We turn to UB, for which we exploit our knowledge about the clusters of r, and the multiplicity of each. More precisely, for each such cluster R, we know the number $\mu(R)$ of roots of r inside R, counted with multiplicity (compare Lemma 17). To derive an adaptive upper bound for $|\mathrm{sres}_k(r, r')|$, we study the possible values of the root product of r.

Definition 19. *Given m' clusters $R_1, \ldots, R_{m'}$ for r, a set $\Psi := \{\psi_1, \ldots, \psi_n\} \subset \bigcup_{i=1,\ldots,m'} R_i$ with $n \geq m'$ is called a* valid root distribution (v.r.d.) *of order n, if the number of elements of V inside R_i is at least one, and at most $\mu_i := \mu(R_i)$. We also define the product*

$$P(\Psi) := \prod_{1 \leq i < j \leq n} (\psi_i - \psi_j)^2.$$

The roots of r obviously define a v.r.d. of order m. We need some additional notation: For two clusters R_i and R_j of r, not necessarily distinct, define d_{ij} to be the maximal distance between a point in R_i to a point in R_j (for $i = j$, d_{ij} is the diameter of R_i).

Proposition 20. *Let $\Psi := \{\psi_1, \ldots, \psi_{m'}\}$ be a v.r.d. of order m'. Then $P(\Psi) \leq \prod_{1 \leq i < j \leq m'} d_{ij}^2$.*

Moreover, we can relate v.r.d.'s of order $n+1$ with v.r.d.'s of order n as follows.

Lemma 21. *Given m' clusters $R_1, \ldots, R_{m'}$ for r, and two v.r.d.'s $\Psi^{(n)} := \{\psi_1, \ldots, \psi_n\}$, $\Psi^{(n+1)} := \{\psi_1, \ldots, \psi_{n+1}\}$ of order n and $n+1$, respectively. Then $P(\Psi^{(n+1)}) \leq \beta^2 P(\Psi^{(n)})$ with*

$$\beta := \max_{i=1,\ldots,m'} \{\prod_{j=1}^{m'} d_{ij} \max\{d_{ij}, 1\}^{\mu_j - 1}\}$$

Proof. Notice that $P(\Psi^{(n+1)})$ can be written as the product of $P(\Psi^{(n)})$ and $\prod_i |\psi_i - \psi_{n+1}|^2$. Let c_i, $i = 1, \ldots, m'$, denote the number of elements among $\psi_1, \ldots, \psi_n$ inside the cluster R_i. By definition, $1 \leq c_i \leq \mu_i$ for all i. Further, let q be such that $\psi_{n+1} \in R_q$. Then, the additional factor $\prod_i |\psi_i - \psi_{n+1}|^2$ can be upper bounded by

$$\prod_{j=1}^{m'} \left(d_{qj}^{c_j}\right)^2 = \prod_{j=1}^{m'} \left(d_{qj} d_{qj}^{c_j - 1}\right)^2 \leq \left(\prod_{j=1}^{m'} d_{qj} \max\{1, d_{qj}\}^{\mu_j - 1}\right)^2. \quad \square$$

Theorem 22. *Define*

$$UB := 2^{\sigma(2D-1)} \cdot e^{D/e} \cdot \beta^2 \cdot \max\{\beta, 1\}^{2(k'-1)} \prod_{1 \leq i < j \leq m'} d_{ij}^2,$$

with σ as in Lemma 4. If $k < k'$, then $\mathrm{sres}_k(r, r') \leq UB$.

Proof. Combining [10, Prop. 3.7] with [3, Prop. 4.27], we obtain the following equation for sres_k:

$$\mathrm{sres}_k(r, r') \quad = \quad \lambda^{2(D-k)-1} \prod_{i=1,\ldots,m} \mathrm{mult}(\phi_i) \cdot P(\Phi),$$

where $\Phi := \{\phi_1, \ldots, \phi_m\}$ are the roots of r, $\mathrm{mult}(\phi_i)$ is the multiplicity of ϕ_i, and λ is the leading coefficient of r.

We bound each factor separately. $\lambda^{2(D-k)-1} \leq 2^{\sigma(2D-1)}$ is obvious, as 2^σ is an upper bound for λ (cf. Lemma 4). For $\prod_i \mathrm{mult}(\phi_i)$, note that this is a product of $D-k$ numbers that sum up to D. One can show that such a product is bound by $\left(\frac{D}{D-k}\right)^{D-k}$, and by maximizing $\left(\frac{d}{x}\right)^x$, one easily verifies that its maximum is $e^{D/e}$.

For the last factor, reorder the roots Φ of r such that ϕ_i lies in cluster R_i, for $i = 1, \ldots, m'$, the other $m - m'$ roots lie in arbitrary clusters. Since Φ is a v.r.d. of order m, $\Phi^{(n)} := \{\phi_1, \ldots, \phi_n\}$ is a v.r.d. of order n for each $m' \leq n \leq m$. Applying Lemma 21 ($m - m'$) times, we obtain that $P(\Phi) \leq \beta^{2(m-m')} P(\Phi^{(m')})$. Now the claim follows by Proposition 20, and by the fact that $\beta^{2(m-m')} = \beta^{2(k'-k)} = \beta^2 \beta^{2(k'-k-1)} \leq \beta^2 \max\{1, \beta\}^{2(k'-k-1)} \leq \beta^2 \max\{1, \beta\}^{2(k'-1)}$. $\square$

Indeed, the bound as defined in Theorem 22 has the properties that we demanded for UB. As just shown, it is an upper bound for $|\mathrm{sres}_k(r, r')|$, if $k < k'$. Moreover, β, as defined in Lemma 21 becomes smaller when the clusters get smaller. If $k = k'$, the diameters d_{ii} all tend to zero, when `Subdivide` is repeatedly called. As each possible value of β contains at least one diameter d_{ii} as a factor, UB tends to zero, since all other quantities are non-increasing.

We next describe the certification phase. From the synchronization phase, the algorithm knows current guesses m'_j, and values LB_j, for each $j = 1,\ldots,N$. Also, it has stored a set of clusters $\mathscr{C}$, which induce precisely m'_j projected clusters under π_j, and the multiplicity of each projected cluster.

For each $j = 1,\ldots,N$, call simultaneously (or alternately) `Guess_k`$_j$ and `Subdivide`. When `Guess_k`$_j$ returns $(k_j^{(p)}, p)$, update the values m'_j and LB_j accordingly, as in the synchronization phase. If m'_j increases, the guess was wrong; switch back to the synchronization phase. After each call of `Subdivide`, update $\mathscr{C}$ and the projected clusters under π_j. If the number of projected clusters has been increased, the guess was wrong; switch back to the synchronization phase. Otherwise, compute UB_j as in Theorem 22. If $LB_j > UB_j$, it is certified that $m'_j = m_j$; proceed with the next j. When all j's have been considered, return the isolating clusters $\mathscr{C}$.

In case of a wrong guess m'_j, the certification phase not just falsifies $m'_j = m_j$, but also increases either m'_j itself, or the number of projected clusters under π_j. Consequently, the certification phase is never called twice for the same guesses $(m'_1,\ldots,m'_N)$. Since the guess can only increase finitely often, this shows that the algorithm eventually terminates. However, we remark again that it is rather unlikely that the algorithm switches back to the synchronization phase at all, as we expect in practice that already the first call of `Guess_k`$_j$ will yield k_j as result.

Remark. The bound $UB = UB_j$ used by the algorithm certainly has room for optimizations. We hint at an alternative bound, based on the following result.

Lemma 23. *For a set $\mathscr{R}$ of projected clusters $R_1,\ldots,R_{m'}$ and maximal distances d_{ij} between R_i and R_j, consider*

$$M(\mathscr{R}) := \max \lambda^{2n-1} \cdot \prod_i \prod_{1\le l\le n_i} \mu_{i,l} \cdot \prod_i d_{ii}^{n_i(n_i-1)} \prod_{1\le i<j\le n} d_{ij}^{n_i n_j}$$

where the maximum is taken over all n ($m' < n \le D$) and all possible values $n_i, \mu_{il} \ge 1$, $i = 1,\ldots,m'$, and $l = 1,\ldots,n_i$ such that $\sum_i n_i = n$ and $\sum_l \mu_{il} = \mu(R_i)$. If $k < k'$, $M(\mathscr{R})$ constitutes an upper bound of $|\mathrm{sres}_k(r,r')|$.

The estimation follows by considering a valid root distribution $\Psi^{(n)}$ of order n such that within each R_i there are exactly n_i elements of $\Psi^{(n)}$ which we assign multiplicities μ_{il}. We omit a more detailed proof for brevity.

For an improved UB, each factor of the product is upper-bounded. The factor $\prod_i \prod_l \mu_{i,l}$ can easily be bounded by $(D/n)^n$. The factor involving the d_{ij}'s leads to a quadratic convex optimization problem in the variables n_i whose matrix becomes diagonal dominant for sufficiently small clusters. Obviously, such an UB is computationally more involved compared to Theorem 22, but the sharper upper bound might amortize this additional cost.

7 Conclusion and further work

We have described a novel certification scheme that allows to certify a collection of regions in $\mathbb{C}^N$ as isolating for the solutions of a zero-dimensional polynomial system over $\mathbb{C}^N$. Our approach combines the advantages of subdivision and modular symbolic computation. The output is certified by homotopy arguments and bounds on subresultants. We emphasize that an exact evaluation of resultants or Groebner bases is not necessary, that is, we only perform modular computations without lifting the elimination polynomials to $\mathbb{Z}$. Thus, all (intermediate) results are kept handy during the computation. At the same time, the performance of the proposed method adaptively depends on

several magnitudes in the algorithm, such as the separation of roots and the size of $\mathrm{sres}_{k_j}(r_j, r'_j)$, instead of using worst-case bounds for them. To the best of our knowledge, only theoretical worst case bounds have been studied [5, 6, 8] so far. With respect to practical efficiency, these bounds are not applicable which results from the fact that, in all situation, the worst case scenario has to be taken into account. We consider our method as the first approach to introduce an adaptive root separation bound. On the one hand our method processes the exact information given by the integer coefficients of the polynomial system by the use of modular computation, and on the other hand, our bound directly depends on the actual geometric situation given by the roots of the system.

Our exposition in this paper is comprehensive in its essence, but does not mention all optimizations that an actual implementation should take care of. For instance, we expect that a more careful choice of the bound UB_j, as suggested at the end of Section 6.2, speeds up the certification phase. Recently, it was shown [23] that, for isolating the complex roots of a univariate polynomial of degree N and bitsize L, a subdivision approach based on centered box evaluation is quite effective. In the corresponding paper, the authors introduce a method called CEVAL that uses centered box evaluation as an exclusion predicate. They showed that the width of the subdivision tree does not exceed $O((N \log N)^2)$ boxes at each subdivision. Applying a different approach to certify the existence of exactly one root within a region, it was also proven that the overall method requires only $\tilde{O}(N^4 L^2)$ bit operations which matches the costs of most effective and exact methods for real root isolation. For higher dimensional systems no such results are available yet, but we are convinced that the techniques from the one dimensional case also apply to the more general setting. Our future research will concentrate on such an analysis.

For an efficient implementation we propose to use fast numerical methods to *find* approximations of the roots. As these methods do not provide any guarantees for their output we consider the role our subdivision methods as crucial in order to *certify* that a region contains (a certain number of) roots. We see a hybrid approach combining a fast numerical solver with our method: Therein, the numerical solver serves as a fast tool to achieve good approximations of the solutions. From our subroutines *Subdivide* and *Mult_of_clusters*, based on subdivision, we determine the exact number of roots within each of the obtained regions and check whether all solutions are captured. If this test fails, the numerical method is restarted with increased arithmetic precision arithmetic. With our *certification phase*, it is possible to verify that the regions are isolating. Thus, assuming that the numerical method determines arbitrary good approximations of all solutions for some sufficiently large precision, this shows the feasibility of a hybrid certified method to isolate all roots.

We further remark that all proposed methods are perfectly suited for parallel computations. For the modular computations this is due to the fact that unhandy terms are avoided and that it is possible to run a large number of distinct modular computations in parallel. The latter holds for the subdivision routines as well, since distinct clusters can be examined independently. We plan to implement and benchmark our algorithm to answer the question whether these advantages lead to measurable effects also in practice.

Our algorithm requires the solution set of the input system to be zero-dimensional. Furthermore, its resultants must be computable over a prime field $\mathbb{Z}_p$. We achieve this by representing resultants as determinants of Macaulay matrices, but for $N \geq 3$ this might fail in unfortunate cases. This constitutes the only obstacle for our algorithm to be complete for higher dimensions. A natural question is whether our ideas also apply to non zero-dimensional systems, and whether a variation of the proposed method can guarantee to solve the system in all cases.

References and Notes

[1] L. Alberti, B. Mourrain, and J. Wintz. Topology and Arrangement Computation of Semi-Algebraic Planar Curves. *Computer Aided Geometric Design*, 25(8):631–651, 2008.

[2] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Computer Science and Applied Mathematics. New York: Academic Press Inc., 1983.

[3] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer, 2nd edition, 2006.

[4] H. Brönnimann, I. Z. Emiris, V. Y. Pan, and S. Pion. Computing exact geometric predicates using modular arithmetic with single precision. In *SCG '97: Proc. of the 13th Ann. Symp. on Computational Geometry*, pages 174–182. ACM press, 1997.

[5] W. D. Brownawell and Chee K. Yap. Lower bounds for zero-dimensional projections. In *Proc. Int'l Symp. Symbolic and Algebraic Comp. (ISSAC'09)*, page To appear, 2009. KIAS, Seoul, Korea, Jul 28-31, 2007. DOI: http://doi.acm.org/10.1145/1277548.1277562. In press, Journal of Symbolic Computation.

[6] M. Burr, S.W. Choi, B. Galehouse, and C. Yap. Complete subdivision algorithms, II: Isotopic meshing of singular algebraic curves. In *Proc. Int'l Symp. Symbolic and Algebraic Computation (ISSAC'08)*, pages 87–94, 2008. Hagenberg, Austria. Jul 20-23, 2008.

[7] Michael Burr, Sung Woo Choi, Benjamin Galehouse, and Chee K. Yap. Complete subdivision algorithms, II: Isotopic Meshing of Singular Algebraic Curves. In *ISSAC'08:Proc. of the 2008 Int. Symp. on Symbolic and Algebraic Computation*, pages 87–94. ACM press, 2008.

[8] Jin-San Cheng, Xiao-Shan Gao, and Chee K. Yap. Complete numerical isolation of real zeros in general triangular systems. In *Proc. Int'l Symp. Symbolic and Algebraic Comp. (ISSAC'07)*, pages 92–99, 2007. Waterloo, Canada, Jul 29-Aug 1, 2007. DOI: http://doi.acm.org/10.1145/1277548.1277562. In press, Journal of Symbolic Computation.

[9] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer, New-York, 1998.

[10] A. Eigenwillig. *Real Root Isolation for Exact and Approximate Polynomials Using Descartes' Rule of Signs*. PhD thesis, Saarland University, Saarbrücken, Germany, 2008.

[11] A. Eigenwillig, M. Kerber, and N. Wolpert. Fast and Exact Geometric Analysis of Real Algebraic Plane Curves. In *ISSAC'07: Proc. of the 2007 Int. Symp. on Symbolic and Algebraic Computation*, pages 151–158. ACM press, 2007.

[12] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Disrcriminants, Resultants and Multidimensional Determinants*. Birkhauser, 1994.

[13] Laureano Gonzalez-Vega and Ioana Necula. Efficient Topology Determination of Implicitly Defined Algebraic Plane Curves. *Comput. Aided Geom. Des.*, 19(9):719–743, 2002.

[14] Irina Voiculescu Huahao Shou, Ralph Martin and et al. Affine arithmetic in matrix form for polynomial evaluation and algebraic curve drawing. *Progress in Natural Science*, 12 (1):77–81, 2002.

[15] L. Kaup and B. Kaup. *Holomorphic Functions of Several Variables*. de Gruyter, 1983.

[16] S. Krishnan, M. Foskey, T. Culver, J. Keyser, and D. Manocha. PRECISE: efficient multiprecision evaluation of algebraic roots and predicates for reliable geometric computation. In *SCG '01: Proc. of the 17th Ann. Symp. on Computational Geometry*, pages 274–283. ACM Press, 2001.

[17] F.S. Macaulay. On some Formula in Elimination. *Proceedings of London Mathematical Society*, pages 3–27, 1902.

[18] R. E. Moore. *Methods and applications of interval analysis*, volume 2 of *SIAM Studies in Applied Mathematics*. SIAM, 1979.

[19] Bernard Mourrain and Jean-Pascal Pavone. Subdivision methods for solving polynomial equations. Technical report, INRIA - Sophia Antipolis, 2005.

[20] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Pres, 1990.

[21] Victor Y. Pan. Sequential and parallel complexity of approximate evaluation of polynomial zeros. *Computers Math. Applic.*, 31(12):97–138, 1996.

[22] S. Petitjean. Algebraic Geometry and Computer Vision: Polynomial Systems, Real and Complex Roots. *J. Math. Imaging Vis.*, 10(3):191–220, 1999.

[23] Michael Sagraloff and Chee K. Yap. An efficient and exact subdivision algorithm for isolating complex roots of a polynomial and its complexity analysis. Draft, unpublished, 2009.

[24] Arnold Schönhage. The fundamental theorem of algebra in terms of computational complexity. *Manuscript, Department of Mathematics, University of Tübingen*, 1982.

[25] B. T. Smith. Error Bounds for Zeros of a Polynomial Based Upon Gerschgorin's Theorems. *J. ACM*, 17(4):661–674, 1970.

[26] A. J. Sommese and C. W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engeneering and Science*. World Scientific, Singapore, 2005.

[27] B. Sturmfels. *Solving systems of polynomial equations*, volume 97 of *Regional conference series in mathematics*. AMS, Providence, RI, 2002.

[28] B. Sturmfels and A. Zelevinsky. Multigraded Resultants of Sylvester Type. *J. Algebra*, 163(1):115–127, 1994.

[29] C. K. Yap. *Fundamental Problems in Algorithmic Algebra*. Oxford University Press, 2000.

A Practical Method for Floating-point Gröbner Basis Computation

Tateaki Sasaki

University of Tsukuba
Ten-Oudai 1-1-1, Tsukuba-shi, Ibaraki 305-8571, Japan
sasaki@math.tsukuba.ac.jp

Abstract

Computing Gröbner bases with inexact coefficients is eagarly desired in industrial applications, but the computation with floating-point numbers is quite unstable if performed naively. In previous papers, the present author clarified that large main-term cancellations occur frequently making the computation unstable, and he proposed a method of removing the harm of exact cancellations. However, the estimation of the amounts of inexact cancellations is very rough and never satisfactory. The inexact cancellations are due to the ill-conditionedness of the system and damage the accuracy of the output system, hence its estimation is very important. In this paper, we propose a new practical method of estimating the amounts of inexact cancellations simply and accurately, which allows us to estimate the accuracy loss in the Gröbner basis computed. Furthermore, we propose two ideas to reduce the amounts of exact and ineaxct main-term cancellations, with preliminary experiments.

1 Introduction

By "floating-point Gröbner basis" we mean a Gröbner basis of polynomial ideal computed with floating-point numbers. There are two kinds of floating-point Gröbner bases. The first kind is that the coefficients of the input polynomials are exact (say algebraic numbers or transcendental numbers) but we utilize the floating-point numbers for some reasons. The second kind is that the coefficients of input polynomials are inexact hence we inevitably express the coefficients by floating-point numbers. If the numerical errors increase during the computation, we can replay the computation with higher precision for the first kind, however, for the second kind, we must devise to preserve the initial accuracies of the given polynomials as far as possible. In this paper, we deal with the second kind.

The first kind of floating-point Gröbner bases were studied by Shirayanagi and Sweedler [13, 14, 15]. The second kind of floating-point Gröbner bases were studied by Stetter [16], Kalkbrener [5], Fortuna, Gianni and Trager [3], Traverso and Zanoni [21], Traverso [20], Weispfenning [22], Kondratyev, Stetter and Winkler [7], Gonzalez-Vega, Traverso and Zanoni [4], Stetter [18], Bodrato and Zanoni [1], and so on. In spite of these studies, computation of floating-point Gröbner bases of the second kind has been a serious problem until recent years; the computation was so unstable in most cases if performed naively. This seriousness forced Mourrain to propose the so-called "border bases" [8, 9]. Furthermore, recently, Suzuki [19] and Nagasaka [10] proposed to compute Gröbner bases by reducing large numerical matrices by Gaussian elimination. The border basis is a set of monomials surrounding the monomials which constitute a basis of the residue class ring of the ideal, hence it is definable only for 0-dimensional ideals. The author is wondering whether the border bases can be computed stably when the input polynomials are inexact. The stability of linear algebra algorithm is not analyzed yet.

Work supported in part by Japan Society for the Promotion of Science under Grants 19300001.

Why the floating-point Gröbner basis computation is so unstable? Shirayanagi [13] pointed out the appearance of *fully-erroneous terms* and Sasaki and Kako [11] pointed out the occurrence of *main-term cancellations*. In the computation with floating-point numbers, subtraction of mathematically the same terms often gives a term with coefficient of no significant bit, and we call such a term fully-erroneous term. In [11], Sasaki and Kako clarified a simple mechanism of exact cancellations of terms in the Gröbner basis computation. If there appears a polynomial of small or large leading term then, in later stages of the computation, large main-term cancellations may occur causing large errors. The fully-erroneous terms can be removed simply and efficiently by expressing the input coefficients either by intervals [13] or by "effective floating-point numbers" ("efloats" in short) [11]. Here, the efloats are floating-point numbers proposed by Kako and Sasaki [6] to detect the cancellation errors approximately but efficiently. The problem is how to protect the initial accuracy from large cancellation errors.

Sasaki and Kako classified the main-term cancellations into two types, exact and inexact cancellations. In [12], Sasaki and Kako found that if we increase the precision then we can remove the harm of the exact cancellations, which is the *high-precision method*; we will explain this method in Sect. 2. However, the method is powerless for the inexact cancellations. Sasaki and Kako also proposed a method to estimate the amounts of inexact cancellations occurred on the coefficients. However, their method is complicated and unsatisfactory in that the estimation is quite rough and theoretically incomplete: the method forsees the appearance of large exact cancellations and performs the polynomial subtraction by removing the main-terms which cancel exactly in the subtraction, but not all the mechanisms of exact cancellations have been clarified yet. On the other hand, the inexact cancellation is very important because its amount is directly related with the accuracy of the output system. Furthermore, we must discard intermediate polynomials the accuracies of which are almost lost by the inexact cancellations; this means that the floating-point Gröbner bases are inevitably related with "approximate Gröbner bases". Therefore, the problems are how to estimate the amounts of inexact cancellations accurately and how to reduce them efficiently.

In this paper, we propose a simple and practical method of estimating the amounts of inexact cancellations occurring on the coefficients of intermediate as well as final polynomials fairly accurately. In our new method, the high-precision method is combined with a "marking" method which we propose in this paper. Furthermore, we propose two devices to reduce the amounts of exact and inexact cancellations.

2 Exact cancellation and high-precision method

By F, G, etc., we denote polynomials in $\mathbb{C}[x, y, \ldots, z]$; the coefficients are actually represented by floating-point numbers. By $\|F\|$ we denote the *norm* of F; we employ the infinity norm. The term (monomial) with no coefficient is called the *power product*. By $\mathrm{lt}(F)$, $\mathrm{lc}(F)$ and $\mathrm{rt}(F)$, we denote the *leading term* (monomial), the *leading coefficient*, and the *rest terms*, of F, respectively, with respect to a given order $\succ$: $F = \mathrm{lt}(F) + \mathrm{rt}(F)$, $\mathrm{lt}(F) \succ \mathrm{rt}(F)$. By $\mathrm{Spol}(F, G)$ and $\mathrm{Lred}(F, G)$, we denote the *S-polynomial* of F and G and the *leading-term reduction* of F by G, respectively.

In this paper, *we restrict the reduction of polynomials only to the leading term reduction*: we compute the Gröbner bases by constructing S-polynomials and performing the leading-term reductions successively. This restriction is essential in our theoretical analysis. If we need the reduced Gröbner basis then we perform the reduction of non-leading terms after

computing an unreduced Gröbner basis.

No matter how the exact cancellation is caused, Sasaki and Kako showed in [12] that the following method protects the initial accuracy from the cancellation errors, so long as the cancellation is exact.

High-precision method Let the accuracy of polynomials of the given initial basis be $\varepsilon_{\text{init}}$ ($\varepsilon_{\text{init}} \ll 1$). Let C_{exct} be the maximum of the exact cancellations occurred on the coefficients of floating-point Gröbner basis computed. Then, initially, set the computational precision ε_{cal} to $\varepsilon_{\text{init}}/C_{\text{exct}}$, convert all the coefficients of the input polynomials to efloats or intervals of precision ε_{cal}, and perform the Gröbner basis computation (Buchberger's procedure). After computing the Gröbner basis, recover the original precision.

Of course, C_{exct} is not known before the computation but we can know it by increasing the precision as $\varepsilon_{\text{cal}} = 10^{-30} \Rightarrow 10^{-60} \Rightarrow 10^{-90} \Rightarrow \cdots$, for example.

We explain why this method protects the initial accuracy even if large main-term cancellations occur. Consider, for example, the computation of $c+d-c$, where $c \simeq 10^{10}$ and $d \simeq 1$ and that $c+d$ is computed first, hence the exact cancellation of about 10^{10} occurs in the computation of $(c+d)-c$. Suppose that both c and d have relative accuracy 10^{-15} initially; see Fig. 1. In the high-precision method, c and d are converted to numbers of 30 decimal figures, for example, among which 15 figures padded at tails are completely meaningless. However, all the 30 figures of c and d are treated by the computer as definite numbers with formal relative errors 10^{-30}. When c and d are added, $c+d$ becomes a number of formal error about 10^{-20}. After subtracting c from $c+d$, the result becomes a number of magnitude 1 with formal relative error about 10^{-20}, because c and $-c$ cancel exactly. Hence the initial accuracy 10^{-15} of d is preserved in the computation.

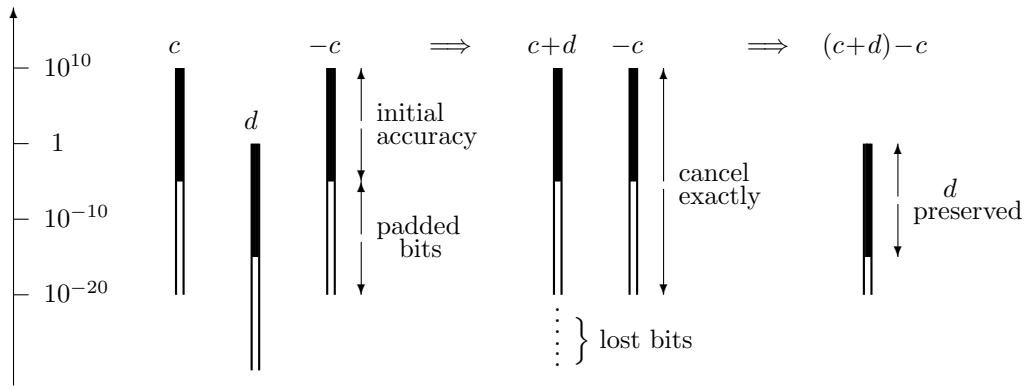


Fig. 1: Computation of $c+d-c$, $|c| \gg |d|$, by high-precision method

3 On inexact cancellation

The high-precision method is useless for errors caused by inexact cancellation; the mechanism illustrated by Fig. 1 is not applicable to inexact cancellations. We show a simple example in which inexact cancellations occur.

Example 1 (inexact cancellation) Let P_1, P_2, P_3 be as follows.

$$\left\{\begin{array}{lcl} P_1 &=& 57/56\,x^2y + 68/67\,xz^2 - 79/78\,xy + 89/88\,x \\ P_2 &=& xyz^3 - xy^2z + xyz \\ P_3 &=& 56/57\,xy^2 - 67/68\,yz^2 + 78/79\,y^2 - 88/89\,y \end{array}\right\} \tag{3.1}$$

We convert the coefficients into double-precision floating-point numbers, which introduces relative errors of 2×10^{-16} into the coefficients. Then, we compute a Gröbner basis w.r.t. total-degree order with 10^{-30}-precision floating-point numbers, obtaining

$$\left\{\begin{array}{lcl} P_1 \text{ and } P_3 \text{ are unchanged}, \quad P_2 \to P_2' \text{ (no cancellation)}, \\ P_6 &=& y^2z^2 - \underline{2.995436947732552644538319700370}\,xy^2 \\ && - \underline{1.002078216512374825767495109674}0\,y^3 \\ && + \underline{1.998325469173724514019288562156}0\,xy \\ && + \underline{1.003521717256414556431326513500}\,y^2, \\ P_7 &=& xz^2 - \underline{1.764316342370426661429391997320}\text{e}{-3}\,yz^2 + \cdots \end{array}\right.$$

The underlines show the correct figures found by comparing with the computation over $\mathbb{Q}$.

We see that cancellations of about 10^4 occurred on several coefficients even in high-precision computation. The computation was performed as follows: first, $P_4 := \mathrm{Spol}(P_1, P_3)$ is computed, then P_2 is reduced by P_4, and we face the following approximate linear dependence in this reduction.

$$\|56/57\,yzP_1 - 57/56\,xzP_3 - 2P_2\| = 0.000041.$$

This approximate linear dependence causes an inexact cancellation of about 10^4, of the main terms of P_2 and P_4. ♢

Let $\{F_1, \cdots, F_r\}$ be an initial basis. Example 1 suggests that the inexact cancellation is caused by approximate linear dependences among polynomials concerned. On the other hand, any polynomial P appearing in Buchberger's procedure can be expressed as $P = a_1F_1 + \cdots + a_rF_r$, where $a_i \in \mathbb{C}[x, y, \ldots, z]$ $(i = 1, \ldots, r)$. We call $(a_1, \ldots, a_r)$ *syzygy* for P, and we have an approximate linear dependence if $\|P\| \ll \max\{\|a_1F_1\|, \cdots, \|a_rF_r\|\}$. Therefore, we investigate relationship between inexact cancellations and syzygies.

Let $\{F_1', \cdots, F_s'\}$ be an intermediate basis (may be the initial basis) computed by Buchberger's procedure, and assume that the polynomial P is constructed as $P = T_iF_i' - T_jF_j'$, where T_i and T_j are monomials (if $P = \mathrm{Lred}(F_i', F_j')$ then we set $T_i = 1$). Let syzygies for F_i' and F_j' be $(a_{i1}', \ldots, a_{ir}')$ and $(a_{j1}', \ldots, a_{jr}')$, respectively. Then, $(a_1, \ldots, a_r)$ is computed by the following formula.

$$(a_1, \ldots, a_r) = T_i\,(a_{i1}', \ldots, a_{ir}') - T_j\,(a_{j1}', \ldots, a_{jr}'). \tag{3.2}$$

Exact cancellation may occur accidentally, for example, in the subtarction $ac - bd$, with a, b, c, d numbers satisfying $ac = bd$. However, in the Gröbner basis computation, most exact cancellations are caused systematically. In [11] and [12], Sasaki and Kako clarified the following mechanism of the exact cancellation.

Known mechanism of exact cancellation Systematic exact cancellation of terms in Buchberger's procedure is caused typically, as follows. Consider $P_1 = \mathrm{Op}(F_i', F_k')$ and $P_2 = \mathrm{Op}(F_j', F_k')$, $i \neq j \neq k$, where Op is either Spol or Lred, hence $P_1 =$

$b_1F'_i + c_1F'_k = b_1\,\mathrm{rt}(F'_i) + c_1\,\mathrm{rt}(F'_k)$, $P_2 = b_2F'_j + c_2F'_k = b_2\,\mathrm{rt}(F'_2) + c_2\,\mathrm{rt}(F'_k)$. As the computation proceeds, P_1 and P_2 may be changed to

$$P'_1 = c'_1F'_k + b'_{11}F'_1 + \cdots + b'_{1s}F'_s, \qquad P'_2 = c'_2F'_k + b'_{21}F'_1 + \cdots + b'_{2s}F'_s, \tag{3.3}$$

where $\mathrm{lt}(\cdots \mathrm{lt}(F'_k)\cdots)$ was canceled in both P'_1 and P'_2 but $Q = \mathrm{rt}(\cdots \mathrm{rt}(F'_k)\cdots)$ remains uncenceled. At some later step of computation, the multiples of Q in the above $c'_1F'_k$ and $c'_2F'_k$ cancel exactly in the computation of $P = \mathrm{Spol}(P'_1, P'_2)$ or $P = \mathrm{Lred}(P'_1, P'_2)$.

Although the mechanism of exact cancellation is not fully clarified yet, the above mechanism is quite general. Therefore, we consider the inexact cancellations in the Gröbner basis computation, by assuming that all the systematic exact cancellations are caused by the above mechanism.

Conjecture 1 *Let polynomial P be either $P = \mathrm{Spol}(P'_1, P'_2)$ or $P = \mathrm{Lred}(P'_1, P'_2)$, and assume that the exact cancellation occurs in the computation of P, as specified by the above mechanism. Let $(a_1, \ldots, a_r)$ be the syzygy for P: $P = a_1F_1 + \cdots + a_rF_r$. Then, the exact cancellations occur in the corresponding terms of $a_1, \ldots, a_r$, too.*

Corollary 1 *If the exact cancellation occurred in P is the main-term cancellation, then almost the same amounts of cancellations occur in the computation of $a_1, \ldots, a_r$.*

Proof With notations in the above mechanism, P can be expressed as $P = T_1P'_1 - T_2P'_2$, where T_1 and T_2 are monomials. By assumption, $T_1c'_1Q$ and $T_2c'_2Q$ cancel each other exactly, which means that $T_1c'_1F'_k$ and $T_2c'_2F'_k$ cancel each other exactly. Expressing $F'_1, \ldots, F'_s$ by $F_1, \ldots, F_r$, we see that there must occur exact cancellations among $a_1, \ldots, a_r$, which correspond to the cancellation of $T_1c'_1F'_k$ and $T_2c'_2F'_k$.
The corollary is a direct consequence of the conjecture. ♢

By Conjecture 1 and its corollary, we characterize the inexact cancellation as follows.

Characterization of inexact cancellation Let P and $(a_1, \cdots, a_r)$ be defined as above. The inexact cancellation occurred on P is characterized by the syzygy for P: if

$$\frac{\max\{\|a_1F_1\|, \cdots, \|a_rF_r\|\}}{\|P\|} \;=\; C_{\mathrm{inxt}} \gg 1, \tag{3.4}$$

then the inexact cancellation of amount C_{inxt} has occurred on P. ♢

4 A new method: high-precision and marking method

In this section, we propose a new practical and stable method for floating-point Gröbner basis computation. Before describing the method, we specify the conditions we are given and point out problems we must solve.

- Each coefficient of the input system contains an error, and we assume that we know the amount of error. Let $\varepsilon_{\mathrm{init}}$ be the accuracy of the coefficients of the initial system: $\varepsilon_{\mathrm{init}}$ is an average value of $|\mathrm{error}(c_i)|/|\mathrm{value}(c_i)|$ $(i = 1, 2, \ldots)$, where c_i is the i-th coefficient of the input system. If large inexact cancellations occur on an intermediate

polynomial and its initial accuracy is almost lost then the polynomial is meaningless hence we must eliminate the polynomial. In particular, if the accuracy of the leading coefficient has been lost fully then we must discard the leading term. Therefore, we are necessary to estimate the amounts of inexact cancellations occurred on the coefficients of not only output polynomials but also all the polynomials appearing in the computation.

- The high-precision method does not reduce the amounts of exact cancellations, and efloats allow us to detect only the sum of exact and inexact cancellations in each coefficient. We can estimate the amount of inexact cancellation occurred on each polynomial by (3.4), but the computation of syzygies is heavy, and the syzygies do not show us the inexact cancellation occurred on each coefficient. We need some clever device to estimate the inexact cancellation occurred on each coefficient accurately.

The above first point means that we are inevitably computing "approximate Gröbner basis" if the input coefficients are inexact.

Our new method, which we call **high-precision and marking method**, is based on the following three devices; we introduce "marks" to solve the second problem.

1. We remove the harm of exact cancellations by the high-precision method; we convert all the coefficient of the input system to high-precision efloats or intervals. Let $\varepsilon_{\rm cal}$ be the computational precision, and let $C_{\rm total}$ be the maximum of the total cancellations (exact + inexact) occurred on the coefficients of output system. If

$$\varepsilon_{\rm cal} < \varepsilon_{\rm init}/C_{\rm total}, \tag{4.1}$$

then $\varepsilon_{\rm cal}$ is sufficient for the computation. If $\varepsilon_{\rm cal}$ is found to be insufficient then we increase the precision and replay the Gröbner basis computation.

2. We prepare two initial systems which are marked differently, and compare two Gröbner bases computed from two initial systems. Here, the marking is done as follows. Consider an initial coefficient c_0 with an error r_0. The mark to this coefficient is a tiny number of magnitude $r_0/|c_0|$ and either added to or subtracted from c_0. By this, a bit of c_0 at the head position of r_0 is changed (marked). After giving marks, we convert the coefficients of the input system into high-precision numbers, as specified above.

3. Let $\Phi_0 = \{F_1, \cdots, F_r\}$ and $\Phi'_0 = \{F'_1, \cdots, F'_r\}$ be two initial systems with high-precision coefficients, where $F_1, \ldots, F_r$ are marked in one way while $F'_1, \ldots, F'_r$ are marked in another way. Hence, mutually corresponding coefficients in Φ_0 and Φ'_0 are the same only in their leading bits above the marks. Suppose $\varepsilon_{\rm cal}$ satisfies (4.1), and consider mutually corresponding polynomials P and P' appearing in intermediate bases started from Φ_0 and Φ'_0, respectively. Let c and c' be mutually corresponding coefficients of P and P', respectively. We divide c and c' into two parts, as follows.

$$c = \bar{c} + d, \quad c' = \bar{c} + d', \tag{4.2}$$

where d and d' are different in their leading bits and the lower bits of $\bar{c}$ starting from the leading bits of d and d' are 0. Then, we can regard that d and d' are errors while $\bar{c}$ is accurate down to the bit just before the leading bits of d and d'. Then, the amount of inexact cancellation occurred on c is given by $\max\{|d|, |d'|\}/(\varepsilon_{\rm init}\bar{c})$.

Figure 2 illustrates the high-precision and marking method. The c_0 and c_0' are mutually corresponding coefficients (normalized to 1.0) of initial systems marked differently at $\varepsilon_{\text{init}}$: symbols $\circ\circ\circ$ and $***$ show different figures padded to make their precision ε_{cal}. Although padded figures are different, the computations are performed almost the same because the computer treats c_0 and c_0' as numbers of accuracy ε_{cal}. The c and c' are mutually corresponding coefficients in intermediate or the final systems marked differently. Many tail figures of c and c' will be lost by exact cancellations and some leading figures will be lost by inexact cancellations. Since the same inexact cancellation must occur on c and c', we can estimate the amount of inexact cancellation by dividing c and c', as in (4.2).

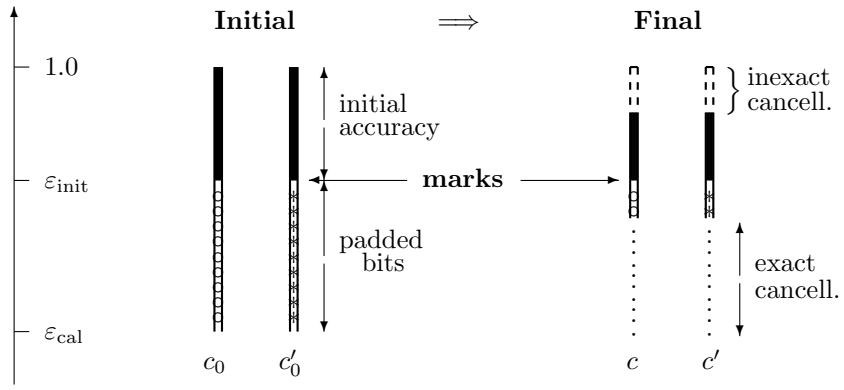


Fig. 2: Illustration of high-precision and marking method

Example 2 (performance of high-precision and marking method)
We apply the high-precision and marking method to the system given in Example 1. We put marks as $P \to P + 1.0\text{e}{-13} \times \text{rt}(P)$ in one system and $P \to P' = P - 1.0\text{e}{-13} \times \text{rt}(P)$ in another system (these markings are too simple in practice). Using big-efloats of 10^{-30}-precision, we obtained the following P_6 and P_6'; underlines show figures which are the same in P_6 and P_6'. (Below, $\#\text{BE}[f, e]$ denotes the big-efloat, where f and e are the value-part and the error-part, respecitively. The e was set to $10^{-28} \times f$ initially.)

$$\begin{aligned} P_6 &= y^2z^2 - \#\text{BE}[\underline{2.995436923212796550471880942470},\ 4.0\text{e}{-24}]\, xy^2 \\ &\quad - \#\text{BE}[\underline{1.00207821651237484}34155553029900,\ 2.0\text{e}{-25}]\, y^3 \\ &\quad + \#\text{BE}[\underline{1.998325444653867541}0656723073600,\ 4.0\text{e}{-25}]\, xy \\ &\quad + \#\text{BE}[\underline{1.00352171725641}45740994791026400,\ 2.0\text{e}{-25}]\, y^2, \\ P_6' &= y^2z^2 - \#\text{BE}[\underline{2.995436}9722795529739694661592 80,\ 1.6\text{e}{-24}]\, xy^2 \\ &\quad - \#\text{BE}[\underline{1.00207821651237484}081194349163500,\ 7.7\text{e}{-25}]\, y^3 \\ &\quad + \#\text{BE}[\underline{1.998325}4937208258344069357248680,\ 1.6\text{e}{-24}]\, xy \\ &\quad + \#\text{BE}[\underline{1.00352171725641}43766318913416240,\ 7.7\text{e}{-25}]\, y^2. \end{aligned}$$

Compared with the results in Example 1, where we introduced relative errors of 2×10^{-16}, we observe that inexact cancellations of about 10^4 occurred on the 1st and 3rd coefficients, whici is consistent with those in Example 1. On the other hand, the 2nd and 4th coefficients show that exact cancellations of about 10^3 occurred on these coefficients but the initial accuracy of 2×10^{-16} was almost preserved. ♢

5 Reducing exact and inexact cancellations

In [21], Traverso and Zanoni tested many practical examples and reported that the cancellations of 2^{100} or more occur frequently. The most parts of such big cancellations are exact ones. Therefore, it is strongly desirable to reduce the amounts of exact cancellations, because the smaller the $\varepsilon_{\rm cal}$ is the more time the computation spends. Furthermore, inexact cancellations damage the accuracy of Gröbner basis computed; in fact, if the inexact cancellation is greater than $1/\varepsilon_{\rm init}$ then the Gröbner basis computed is meaningless. Therefore, we want to reduce the amounts of inexact cancellations, too.

Since large main-term cancellations are caused by polynomials of small or large leading terms, we define the abnormality of polynomial as follows.

Definition 1 (abnormality) *Let a polynomial P be $P = cT + P'$, where $cT = \mathrm{lt}(P)$ with $c = \mathrm{lc}(P)$ and we assume $P' \neq 0$. We define* abnormality *of P as follows.*

$$\text{abnormality} = \begin{cases} \|P'\|/|c| & \text{if } |c| \leq \|P'\|, \\ -|c|/\|P'\| & \text{if } |c| > \|P'\|. \end{cases} \tag{5.1}$$

We expect that, by treating polynomials of large |abnormalities| specially, we will be able to reduce the amounts of exact cancellations. We have tested the following three strategies.

Strategy 1: Divide the computation into two stages. In the first stage, the reduction is performed only by polynomials of small |abnormalities|. The reduction by polynomials of large |abnormalities| are done after the termination of the first stage.

Result: The first stage sometimes does not terminate, because many S-polynomials are constructed and not reduced to 0.

Strategy 2: Divide the computation into two stages. In the first stage, construct only S-polynomials by combining polynomials of similar abnormalities. Combining polynomials of non-similar abnormalities is done in the second stage.

Result: One may expect that many abnormal polynomials are reduced to 0 in the first stage by this strategy. Actually, however, many abnormal polynomials survive to the second stage, and large cancellations occur in the second stage.

Strategy 3: Sort existing polynomials in small-to-large order w.r.t. their |abnormalities|. In constructing S-polynomials and in reducing polynomials, take up polynomials in this sorted order.

Result: This strategy succeeds considerably, as Example 3 below shows.

Example 3 (test of strategy 3) Consider the following system which was given in [11]. We compute the Gröbner basis w.r.t. total-degree order, with two strategies for choosing polynomials in the S-polynomial construction and the reduction.

$$\left\{ \begin{array}{lcl} P_1 &=& x^3/10.0 + 3.0x^2y + 1.0y^2 \\ P_2 &=& 1.0x^2y^2 - 3.0xy^2 - 1.0xy \\ P_3 &=& y^3/10.0 + 2.0x^2 \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} P_2 = y^2 \\ P_4 = xy + \cdots \\ P_5 = x^2 + \cdots \end{array} \right\}$$

If we choose the same strategy as for computing Gröbner bases over $\mathbb{Q}$, we will face the exact cancellations of about 10^9. However, with the Strategy 3 mentioned above, we found that the exact cancellations occurred on P_5, P_2, P_4 are 1010, 89, 1010, respectively. Therefore, Strategy 3 is pretty successful for this example. ♢

Next, we consider to reduce the inexact cancellations. As mentioned in Sect. 3, the inexact cancellations are caused by approximate linear dependences among polynomials concerned. However, the dependence often disappears when we change the term order (of course, another dependence may appear). Therefore, our strategy is as follows.

Strategy 4: Compute a Gröbner basis by changing the term order variously, and if a Gröbner basis of high accuracy is obtained then apply the basis changing algorithm to recover the original term order.

Unfortunately, for the system in Example 1, either the lexicographic or the total-degree reverse-lexicographic term-order cannot reduce the inexact cancellations. However, the above strategy works well for the following system.

Example 4 (test of Strategy 4) Let P_1, P_2, P_3 be as follows.

$$P_1 = x^2\,(2yz+1), \qquad P_2 = yz\,(3xz-2), \qquad P_3 = x\,(3xz-2) - (2yz+1).$$

We have computed the Gröbner basis of P_1, P_2, P_3 with respect to two term-orders, using double-precision efloats. Below, (GBtdg) and (GBlex) denote the unreduced Gröbner bases w.r.t. the total-degree and the lexicographic term-orders, respectively.

$$\text{(GBtdg)} \;:\; \begin{cases} P_4 &=& x \;+\#\mathrm{E}[1.3333333333333, 1.01\mathrm{e}{-11}]\,y, \\ P_2' &=& yz + \#\mathrm{E}[0.058139534883721, 1.47\mathrm{e}{-13}]\,x \\ & & \quad + \#\mathrm{E}[0.077519379844961, 1.96\mathrm{e}{-13}]\,y + \#\mathrm{E}[0.5, 2.00\mathrm{e}{-13}]. \end{cases}$$

$$\text{(GBlex)} \;:\; \begin{cases} P_3' &=& yz + \#\mathrm{E}[0.5, 4.0\mathrm{e}{-15}], \\ P_2' &=& x \;- \#\mathrm{E}[4.0, 4.0\mathrm{e}{-15}]\,y^3z^3 - \#\mathrm{E}[2.0, 4.0\mathrm{e}{-15}]\,y^2z^2 \\ & & \quad - \#\mathrm{E}[2.666, 2.66\mathrm{e}{-15}]\,y^2z + yz + \#\mathrm{E}[0.5, 5.0\mathrm{e}{-16}]. \end{cases}$$

We see that inexact cancellations of about 10^4 and 10^2 have occurred in (GBtdg), while almost no cancellation has occurred in (GBlex). Note that even if we reduce P_2' by P_3' in (GBlex), no accuracy loss occurs. ♢

References and Notes

[1] M. Bodrato and A. Zanoni, Intervals, syzygies, numerical Gröbner bases: a mixed study. *Proceedings of CASC2006 (Computer Algebra in Scientific Computing)*: Springer-Verlag LNCS 4194, 64-76, 2006.

[2] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases (F_4). *J. Pure Appl. Algebra*, **139**, 61-88, 1999.

[3] E. Fortuna, P. Gianni and B. Trager, Degree reduction under specialization. *J. Pure Appl. Algebra*, **164**, 153-164, 2001.

[4] L. Gonzalez-Vega, C. Traverso and A. Zanoni. Hilbert stratification and parametric Gröbner bases. *Proceedings of CASC2005 (Computer Algebra in Scientific Computing)*: Springer-Verlag LNCS 3718, 220-235, 2005.

[5] M. Kalkbrener, On the stability of Gröbner bases under specialization. *J. Symb. Comput.*, **24**, 51-58, 1997.

[6] F. Kako and T. Sasaki, Proposal of "effective" floating-point number. Preprint of Univ. Tsukuba, May 1997 (unpublished).

[7] A. Kondratyev, H.J. Stetter and S. Winkler, Numerical computation of Gröbner bases. *Proceedings of CASC2004 (Computer Algebra in Scientific Computing)*, 295-306, St. Petersburg, Russia, 2004.

[8] B. Mourrain, A new criterion for normal form algorithms. *Lec. Notes Comp. Sci.*, **179**, 430-443, Springer, 1999.

[9] B. Mourrain, Pythagore's dilemma, symbolic-numeric computation, and the border basis method. *Symbolic-Numeric Computations (Trends in Mathematics)*, 223-243, Birkhäuser Verlag, 2007.

[10] K. Nagasaka, A Study on Gröbner basis with inexact input. *Lec. Notes Comp. Sci.: Proceedings of CASC2009 (Computer Algebra in Scientific Computing)*, **5743**, 248-258, Kobe, Japan, 2009.

[11] T. Sasaki and F. Kako, Computing floating-point Gröbner base stably. *Proceedings of SNC2007 (Symbolic Numeric Computation)*, 180-189, London, Canada, 2007.

[12] T. Sasaki and F. Kako, Floating-point Gröbner basis comptation with ill-conditionedness estimation. *Proceedings of ASCM2007 (Asian Symposium on Computer Mathematics)*, Singapore, Dec. 2007; see also LNAI 5081, 278-292, Deepak Kapur (Ed.), Springer, 2008.

[13] K. Shirayanagi, An algorithm to compute floating-point Gröbner bases. *Mathematical Computation with Maple V. Ideas and Applications*, Birkhäuser, 95-106, 1993.

[14] K. Shirayanagi, Floating point Gröbner bases. *Mathematics and Computers in Simulation*, **42**, 509-528, 1996.

[15] K. Shirayanagi and M. Sweedler, Remarks on automatic algorithm stabilization. *J. Symb. Comput.*, **26**, 761-765, 1998.

[16] H.J. Stetter, Stabilization of polynomial systems solving with Gröbner bases. *Proceedings of ISSAC'97 (Intern'l Symposium on Symbolic and Algebraic Computation)*, 117-124, ACM Press, 1997.

[17] H.J. Stetter. *Numerical Polynomial Algebra*. SIAM Publ., Philadelphia, 2004.

[18] H.J. Stetter, Approximate Gröbner bases – an impossible concept?. *Proceedings of SNC2005 (Symbolic-Numeric Computation)*, 235-236, Xi'an, China, 2005.

[19] A. Suzuki, Computing Gröbner bases within linear algebra. *Lec. Notes Comp. Sci.: Proceedings of CASC2009 (Computer Algebra in Scientific Computing)*, **5743**, 310-321, Kobe, Japan, 2009.

[20] C. Traverso, Syzygies, and the stabilization of numerical Buchberger algorithm. *Proceedings of LMCS2002 (Logic, Mathematics and Computer Science)*, 244-255, RISC-Linz, Austria, 2002.

[21] C. Traverso and A. Zanoni, Numerical stability and stabilization of Gröbner basis computation. *Proceedings of ISSAC2002 (Intern'l Symposium on Symbolic and Algebraic Computation)*, 262-269, ACM Press, 2002.

[22] V. Weispfenning, Gröbner bases for inexact input data. *Proceedings of CASC2003 (Computer Algebra in Scientific Computing)*, 403-411, Passau, Germany, 2003.

Series Expansion of Multivariate Algebraic Functions at Singular Points – Nonmonic Case –

Tateaki Sasaki[1*] and Daiju Inaba[2]

[1] University of Tsukuba
Ten-Oudai 1-1-1, Tsukuba-shi, Ibaraki 305-8571, Japan
sasaki@math.tsukuba.ac.jp

[2] Mathematics Certification Inst. Japan
Higashi-Kanamachi, Katsushika-ku, Tokyo 125-8602, Japan
inaba@math.tsukuba.ac.jp

Abstract

In a series of papers, we have developed a method of expanding multivariate algebraic functions at their singular points. The method applies the Hensel construction to the defining polynomial of the algebraic function, so we named the resulting series "Hensel series". In [16], we derived a concise representation of Hensel series and clarified several characteristic properties of Hensel series theoretically when the defining polynomial is monic. In this paper, we study the case of nonmonic defining polynomial. We show that, by determining the so-called Newton polynomial suitably, we can construct Hensel series which show reasonable behaviors at zero-points of the leading coefficients and we can derive a representation of Hensel series in the nonmonic case just similarly as in the monic case. Furthermore, we investigate the convergence/divergence behavior and many-valuedness of Hensel series in the nonmonic case.

1 Introduction

Series expansion is a fundamental tool in numerical analysis, such as for solving differential equations [8], for tracing analytic functions numerically [1], and so on. So far, the Taylor expansion, multivariate as well as univariate, has mostly been used in numerical analysis. The Taylor expansion breaks down at singular points, and most numerical algorithms are constructed to avoid the singular points. However, singular points are very important in mathematics; many important mathematical properties are concentrated at singular points. It is strongly desirable to establish numerical analysis at and near the singular points. For this purpose, we need useful and tractable method of series expansion at singular points.

In a series of papers [5, 13, 14, 15, 16, 17, 18], we have developed a method of expanding multivariate algebraic functions into series at singular points, where the algebraic functions are defined as roots of a given multivariate polynomial. Since our method is based on the Hensel construction, we named the series obtained by our method *Hensel series*. For bivariate polynomials, our method computes Puiseux series roots simultaneously and efficiently; for Puiseux series, see [21]. For multivariate polynomials, different kinds of expansions are possible, and the series obtained by our method are very different from multivariate Puiseux series [2, 10, 11].

The Hensel series was used so far to the analytic continuation of algebraic functions via singular points [20], solving multivariate algebraic equations in series forms [17, 18], the analytic factorization of polynomials of more than two variables [6, 7], and so on. For some other researches of utilizing series expansions at singular points, see [3, 4, 9, 12, 19].

Work supported in part by Japan Society for the Promotion of Science under Grants 19300001.

Let $F(x, \boldsymbol{u}) \stackrel{\text{def}}{=} F(x, u_1, \ldots, u_\ell)$ be a given multivariate polynomial, and let $f_n(\boldsymbol{u})$ be its leading coefficient w.r.t. x. The algebraic functions are affected strongly and delicately by $f_n(\boldsymbol{u})$. If the point $\boldsymbol{u}$ approaches a zero-point of $f_n(\boldsymbol{u})$ then at least one algebraic function goes to infinity, and if $\boldsymbol{u}$ approaches another zero-point then another algebraic function may go to infinity. One will think that generalization to nonmonic case is straightforward. For example, we can employ a famous transformation which converts any nonmonic polynomial to a monic one. However, with such a transformation, the effect of $f_n(\boldsymbol{u})$ is not fully taken into the Hensel series. We want to treat the nonmonic case so that the effect of leading coefficient is taken into the Hensel series as fully as possible.

In this paper, we propose a reasonable treatment of the leading coefficient. In our method, the so-called Newton polynomial plays an essential role, so we include the leading coefficient into the Newton polynomial. This treatment seems to be peculiar at first glance. However, the Hensel series are expressed in the roots of Newton polynomial and many analytic properties of Hensel series are determined by the roots. Therefore, by including the leading coefficient into the Newton polynomial, the effect of the leading coefficient is taken into Hensel series largely. Furthermore, with our treatment, the Hensel series are represented by the same formula as that in the monic case, and we can clarify the properties of Hensel series simply. It is, however, emphasized that this paper treats only the case that the Newton polynomial is squarefree; treatment of general case is complicated, as [13] shows.

2 Our approach to nonmonic case

Let $F(x, \boldsymbol{u}) \stackrel{\text{def}}{=} F(x, u_1, \ldots, u_\ell) \in \mathbb{C}[x, u_1, \ldots, u_\ell]$ be a given multivariate polynomial. By $\deg(F)$ and $\mathrm{lc}(F)$, we denote the degree and the leading coefficient, respectively, w.r.t. x, of $F(x, \boldsymbol{u})$. We put $\deg(F) = n$ and $\mathrm{lc}(F) = f_n(\boldsymbol{u})$. By $\mathrm{tdeg}(f)$, with $f(\boldsymbol{u}) \in \mathbb{C}[\boldsymbol{u}]$, we denote the *total-degree* of $f(\boldsymbol{u})$ w.r.t. sub-variables $u_1, \ldots, u_\ell$; if $T = c\, u_1^{e_1} \cdots u_\ell^{e_\ell}$, $c \in \mathbb{C}$, then $\mathrm{tdeg}(T) = e_1 + \cdots + e_\ell$. By $\mathrm{ord}(f)$ we denote the order of $f(\boldsymbol{u})$, i.e., the minimum of the total-degrees of terms of $f(\boldsymbol{u})$. Let $\varphi(\boldsymbol{u})$ be an algebraic function defined as a root w.r.t. x, of $F(x, \boldsymbol{u})$: $F(\varphi(\boldsymbol{u}), \boldsymbol{u}) = 0$.

Let $\boldsymbol{s} \stackrel{\text{def}}{=} (s_1, \ldots, s_\ell) \in \mathbb{C}^\ell$ be an expansion point. Without loss of generality, we assume that $F(x, \boldsymbol{u})$ is irreducible hence squarefree. If $f_n(\boldsymbol{s}) \neq 0$ and $F(x, \boldsymbol{s})$ is squarefree then every root of $F(x, \boldsymbol{u})$ w.r.t. x can be expanded into Taylor series in $u_1 - s_1, \ldots, u_\ell - s_\ell$.

Definition 1 (singular point, singular leading coefficient) *We call the expansion point $\boldsymbol{s} \in \mathbb{C}^\ell$ a singular point of algebraic function, or a singular point in short, if $F(x, \boldsymbol{s})$ is not squarefree. If $f_n(\boldsymbol{s}) = 0$ then we say the leading coefficient is singular at $\boldsymbol{s}$.*

Without loss of generality, we assume that the origin $\boldsymbol{u} = \boldsymbol{0}$ is a singular point or $f_n(\boldsymbol{u})$ is singular at the origin, and we consider the series expansion at the origin.

We show the importance of treatment of leading coefficient by a simple example.

Example 1 Let $F_1(x, u, v)$ be as follows ("H.T." denotes higher order terms).

$$F_1(x, u, v) \;=\; [(u+v)\, x - (u^2+v^2)] \cdot [uv\, x^2 - (u^4+v^4)] + \text{H.T.}.$$

If $\deg(\text{H.T.}) \leq 2$ then the root $\varphi(u, v)$ of $F_1(x, u, v)$ w.r.t. x behaves as

$$\varphi(u, v) = \frac{u^2+v^2}{u+v} + \text{H.T.}, \quad \varphi(u, v) = \pm\Big(\frac{u^4+v^4}{uv}\Big)^{1/2} + \text{H.T.} \quad \text{for small } |u| \text{ and } |v|.$$

However, if H.T. of $F_1(x,u,v)$ contains $(u^4+v^4)\,x^3$, for example, the algebraic function $\varphi(u,v)$ does not diverge on lines $u+v=0$, $u=0$ or $v=0$. That is, the algebraic function is affected strongly by the leading coefficient of its defining polynomial. $\diamondsuit$

The key concept in the Hensel construction at a singular point is the Newton polynomial $F_{\rm New}(x,\boldsymbol{u})$; see [17, 18]. In treating nonmonic case, we want to define $F_{\rm New}(x,\boldsymbol{u})$ so that the resulting Hensel series reflect the effects of leading coefficient as fully as possible yet we can treat the Hensel series near the expansion point as simply as possible.

An approach which we adopt in this paper is quite simple, although it seems rather peculiar; we define the Newton polynomial as follows.

Definition 2 (Newton polynomial $F_{\rm New}(x,\boldsymbol{u})$, the nonmonic case) *For each monomial $c\,x^i t^j u_1^{j_1}\cdots u_\ell^{j_\ell}$ of $F(x,t\boldsymbol{u})$, with $c\in\mathbb{C}$ and $j=j_1+\cdots+j_\ell$, plot a dot at the point (i,j) in the (e_x,e_t)-plane. Let $\nu=\mathrm{ord}(f_n)$, and let $\mathcal{L}_{\rm New}$ be a straight line in (e_x,e_t)-plane, such that it passes the point (n,ν) and another dot plotted and that any dot plotted is not below $\mathcal{L}_{\rm New}$. Construct $F_{\rm New}(x,t\boldsymbol{u})$ by summing all the monomials plotted on $\mathcal{L}_{\rm New}$ and by replacing* $\mathrm{lc}(F_{\rm New})$ *by $f_n(\boldsymbol{u})$. (Hence,* $\mathrm{lc}(F_{\rm New})=f_n(\boldsymbol{u})$*.)*

Let the slope of $\mathcal{L}_{\rm New}$ be $-\lambda$ (note the "minus" sign). We have assumed that the origin $\boldsymbol{u}=\boldsymbol{0}$ is a singular point. In drawing figures of $\varphi(\boldsymbol{u})$ near the origin, we should "regularize" $\varphi(\boldsymbol{u})$. The behavior of $\varphi(\boldsymbol{u})$ near the origin is approximately determined by the Newton polynomial: if the slope of $\mathcal{L}_{\rm New}$ is positive (negative) then $\varphi(\boldsymbol{u})$ goes to infinity (zero, resp.) as $(u_1,\ldots,u_\ell)\to(0,\ldots,0)$. Therefore, we define "regularized-root" $\bar\varphi(\boldsymbol{u})$ as follows (below, $|\boldsymbol{u}|$ has the meaning only when we substitute values to all the variables $u_1,\ldots,u_\ell$).

$$\bar\varphi(\boldsymbol{u}) \stackrel{\rm def}{=} |\boldsymbol{u}|^{-\lambda}\varphi(\boldsymbol{u}), \quad |\boldsymbol{u}| \stackrel{\rm def}{=} (|u_1|^2+\cdots+|u_\ell|^2)^{1/2}. \tag{2.1}$$

Note that $\bar\varphi(\boldsymbol{s})\to$ [finite value] as $|\boldsymbol{s}|\to 0$ for most values of $\boldsymbol{s}\in\mathbf{C}^\ell$. We also regularize Hensel series $\phi(\boldsymbol{u})$ to be determined in the next section.

3 Hensel series in a compact representation

First of all, we confine ourselves to discussing the following restricted case.

Assumption A *We assume that $F_{\rm New}(x,\boldsymbol{u})$ is square-free not only exactly but also approximately, i.e., we assume that $F_{\rm New}(x,\boldsymbol{u})$ has no multiple/close roots. Furthermore, we assume that $F_{\rm New}(x,\boldsymbol{u})$ has no numeric root.*

The second condition implies that $F_{\rm New}(x,\boldsymbol{u})$ has no x^m factor $(0<m<n)$. If $F_{\rm New}(x,\boldsymbol{u})$ has numeric roots then the corresponding algebraic functions can be expanded into Taylor series, and we are not interested in such a case.

Let the roots of $F_{\rm New}(x,\boldsymbol{u})$ be $\alpha_1(\boldsymbol{u}),\ldots,\alpha_n(\boldsymbol{u})$, which we often write as $\alpha_1,\ldots,\alpha_n$.

$$F_{\rm New}(x,\boldsymbol{u})=f_n(\boldsymbol{u})(x-\alpha_1(\boldsymbol{u}))\cdots(x-\alpha_n(\boldsymbol{u})), \qquad \alpha_i\neq\alpha_j \ \ (\forall i\neq j). \tag{3.1}$$

The $\alpha_i(\boldsymbol{u})$ is usually an algebraic function.

We define $\widetilde{F}(x,\boldsymbol{u},\eta)$ by introducing an auxiliary variable η, as follows.

$$\begin{cases} F(x,\boldsymbol{u}) \stackrel{\rm def}{=} F_{\rm New}(x,\boldsymbol{u})+F_{\rm h}(x,\boldsymbol{u}), \\ \widetilde{F}(x,\boldsymbol{u},\eta) \stackrel{\rm def}{=} F_{\rm New}(x,\boldsymbol{u})+\eta\,F_{\rm h}(x,\boldsymbol{u}). \end{cases} \tag{3.2}$$

We factor $F_{\rm New}(x,\boldsymbol{u})$ as $F_{\rm New}(x,\boldsymbol{u}) = G_1^{(0)}(x,\boldsymbol{u})\cdots G_r^{(0)}(x,\boldsymbol{u})$, where $G_1^{(0)},\ldots,G_r^{(0)}$ are relatively prime. Then, using $G_1^{(0)},\ldots,G_r^{(0)}$ as initial factors, we perform the Hensel construction of $\widetilde{F}(x,\boldsymbol{u},\eta)$ with modulus η to satisfy

$$\widetilde{F}(x,\boldsymbol{u},\eta) \equiv G_1^{(k)}(x,\boldsymbol{u},\eta)\cdots G_r^{(k)}(x,\boldsymbol{u},\eta) \pmod{\eta^{k+1}}, \qquad k=1,2,\ldots.$$

Choosing the initial factors as $G_1^{(0)} = x-\alpha_1$ and $\tilde{G}^{(0)} = F_{\rm New}(x,\boldsymbol{u})/(x-\alpha_1)$, we obtain

$$\begin{cases} \widetilde{F}(x,\boldsymbol{u},\eta) \equiv G_1^{(k)}(x,\boldsymbol{u},\eta)\cdot\tilde{G}^{(k)}(x,\boldsymbol{u},\eta) \pmod{\eta^{k+1}}, \\ G_1^{(0)}(x,\boldsymbol{u},1) = x-\alpha_1, \quad G_1^{(k)}(x,\boldsymbol{u},\eta) = x-\phi_1^{(k)}(\boldsymbol{u},\eta). \end{cases} \tag{3.3}$$

The $\phi_1^{(\infty)}(\boldsymbol{u},1)$ is the Hensel series corresponding to α_1.

Once the Newton polynomial is defined, the Hensel construction is straightforward. Because of the page limit, we omit the details of deriving a representation of Hensel series, for which the reader can refer to [16].

Theorem 1 *Let $F_{\rm New}(x,\boldsymbol{u})$ be squarefree. Then, the Hensel factors $G_1^{(\infty)}$ and $\tilde{G}^{(\infty)}$ in* (3.3) *are expressed as follows.*

$$G_1^{(\infty)}(x,\boldsymbol{u},\eta) = x-\alpha_1+\sum_{k=1}^{\infty}\eta^k\,\frac{\delta F^{(k)}(\alpha_1,\boldsymbol{u})}{F'_{\rm New}(\alpha_1,\boldsymbol{u})}, \tag{3.4}$$

$$\tilde{G}^{(\infty)}(x,\boldsymbol{u},\eta) = \frac{F_{\rm New}(x,\boldsymbol{u})}{x-\alpha_1}+\sum_{j=2}^{n}\frac{F_{\rm New}(x,\boldsymbol{u})}{(x-\alpha_1)(x-\alpha_j)}\Big(\sum_{k=1}^{\infty}\eta^k\,\frac{\delta F^{(k)}(\alpha_j,\boldsymbol{u})}{F'_{\rm New}(\alpha_j,\boldsymbol{u})}\Big), \tag{3.5}$$

where $\delta F^{(1)} = F_{\rm h}(x,\boldsymbol{u})$ and the k-th order residual $\delta F^{(k)}$ $(k\ge 2)$ is given as follows.

$$\delta F^{(k)}(x,\boldsymbol{u}) = -\sum_{j=2}^{n}\frac{F_{\rm New}(x,\boldsymbol{u})}{(x-\alpha_1)(x-\alpha_j)}\Big(\sum_{k'=1}^{k-1}\frac{\delta F^{(k')}(\alpha_1,\boldsymbol{u})}{F'_{\rm New}(\alpha_1,\boldsymbol{u})}\,\frac{\delta F^{(k-k')}(\alpha_j,\boldsymbol{u})}{F'_{\rm New}(\alpha_j,\boldsymbol{u})}\Big). \tag{3.6}$$

Example 2 (numerical evaluation of Hensel series) Let $F_2(x,u,v)$ be as follows.

$$F_2(x,u,v) = [(u+v)\,x-1]\,[uv\,x^2+(u-v)x+1]+(u^3+v^3)\,x^2+(u+v).$$

The Newton polynomial for $F_2(x,u,v)$ is as follows.

$$F_{2\rm New}(x,u,v) = [(u+v)\,x-1]\,[uv\,x^2+(u-v)x+1].$$

Let α_1 be the root of $F_{2\rm New}(x,u,v)$ corresponding to the factor $[(u+v)\,x-1]$ and let the roots of $F_2(x,u,v)$ be $\varphi_i(u,v)$ $(i=1,2,3)$.

Table I. Numerical evaluation.

u	$\bar{\phi}_1^{(4)}(u,0.1)$	$\bar{\varphi}_i(u,0.1)$ $(i=1,2,3)$		
-0.20	-7.11517	0.696242	$-2.75189 \pm 1.12207\,i$	
-0.15	∞	0.635459	$-3.33738 \pm 1.33045\,i$	
-0.10	∞	0.585786	-3.41421	∞
-0.05	2.46037	0.573904	2.46039	-3.76102
0.00	∞	0.626789	1.59543	∞
0.05	0.590837	0.590885	$0.552400 \pm 1.54123\,i$	
0.10	0.565685	0.565685	$\pm 1.41421\,i$	
0.15	0.559016	0.559010	$-0.324574 \pm 1.41098\,i$	
0.20	0.552161	0.552125	$-0.630106 \pm 1.40194\,i$	

We have computed Hensel series $\phi_1^{(k)}(u,v)$ starting from α_1 up to $k=4$, and compared with algebraic functions $\varphi_i(u,v)$ $(i=1,2,3)$ which we computed exactly by Mathematica. Table I shows values of regularized truncated Hensel series $\bar{\phi}_1^{(4)}(u,v)$ and $\bar{\varphi}_1(u,v)$ $(i=1,2,3)$ in the real range $-0.20 \le u \le 0.20$, where we fixed v to 0.1.

The "convergence domain" of $\bar{\phi}_1^{(4)}(u,v)$ will be shown in Sect. 5 (Figs. 1a and 1b). We see that, in the "convergence domain", the Hensel series agrees with one of the algebraic functions fairly well. Note that the values at $u=-0.05$ show "jumping" of Hensel series among algebraic functions, which we will discuss in Sect. 6. ♢

4 Order estimations near the expansion point

Formula (3.4) shows that the behavior of each Hensel series depends critically on the roots $\alpha_1,\ldots,\alpha_n$. In this section, we order-estimate $\alpha_1,\ldots,\alpha_n$ etc. when $\boldsymbol{u}$ is near the origin. Because of the page limit, we omit an analysis of the case near zero-points of $f_n(\boldsymbol{u})$, for which the reader can request the authors to send a full paper version.

Below, we put $F(x,\boldsymbol{u}) = f_n(\boldsymbol{u})x^n + f_{n-1}(\boldsymbol{u})x^{n-1} + \cdots + f_0(\boldsymbol{u})$, and we define $\|\boldsymbol{u}\|$ to be the Euclidean norm of $\boldsymbol{u}$, $\|\boldsymbol{u}\| \stackrel{\text{def}}{=} (|u_1|^2 + \cdots + |u_\ell|^2)^{1/2}$, after substituting suitable numbers for $u_1,\ldots,u_\ell$. Note that we have $\|\boldsymbol{u}\| \ll 1$ in this section, and that if actual values are substituted for $u_1,\ldots,u_\ell$ then we may have $\|\alpha_i(\boldsymbol{u}) - \alpha_j(\boldsymbol{u})\| = 0$. The following order estimations are for generic $\boldsymbol{u}$. Furthermore, by $o(\|\boldsymbol{u}\|^a)$ we mean either $O(\|\boldsymbol{u}\|^{a+1})$ or less.

4.1 When $\mathrm{ord}(f_n) = 0$

The condition $\mathrm{ord}(f_n)=0$ means that $f_n(\boldsymbol{0}) \neq 0$. Since $f_n(\boldsymbol{u})$ has a nonzero constant term, we obtain the same order-estimations as in the monic case.

Lemma 1 *When* $\mathrm{ord}(f_n)=0$*, we have the following order estimations for small* $\|\boldsymbol{u}\|$*, so long as* $\boldsymbol{u}$ *is not close to the zero-points of* $\alpha_i(\boldsymbol{u})-\alpha_j(\boldsymbol{u})$ $(\forall i \neq j)$*.*

$$\|\alpha_1\|,\ \|\alpha_1-\alpha_j\| \ = \ O(\|\boldsymbol{u}\|^\lambda) \quad (j=2,\ldots,n), \tag{4.1}$$

$$\|F'_{\mathrm{New}}(\alpha_i,\boldsymbol{u})\| \ = \ O(\|\boldsymbol{u}\|^{(n-1)\lambda}) \quad (i=1,\ldots,n), \tag{4.2}$$

$$\|F_h(\alpha_i,\boldsymbol{u})\| \ = \ o(\|\boldsymbol{u}\|^{n\lambda}) \quad (i=1,\ldots,n). \tag{4.3}$$

Proof For small $\|\boldsymbol{u}\|$, $\alpha_1,\ldots,\alpha_n$ are determined mostly by the homogeneous parts of $F_{\mathrm{New}}(x,\boldsymbol{u})$ (that is, we can discard the higher order terms of $f_n(\boldsymbol{u})$ when $\|\boldsymbol{u}\|$ is small). Hence, by Assumption A, we obtain (4.1) and (4.2) at once. Since each coefficient of $F_{\mathrm{h}}(x,\boldsymbol{u})$ is of higher order w.r.t. $u_1,\ldots,u_\ell$ than the corresponding coefficient of $F_{\mathrm{New}}(x,\boldsymbol{u})$ by at least 1, we obtain (4.3). ♢

4.2 When $\mathrm{ord}(f_n) = \nu > 0$ and $\|f_n(\boldsymbol{u})\| = O(\|\boldsymbol{u}\|^\nu)$

Under the Assumption A, condition $\mathrm{ord}(f_n)=\nu>0$ means that $\mathrm{ord}(f_i)>0$ for $i \ge 1$ so long as $f_i \neq 0$; $\mathrm{ord}(f_0)$ may be 0 or positive. Hence, $f_i(\boldsymbol{0})=0$ for any $i \ge 1$. If $f_i=0$ or $\mathrm{ord}(f_i)$ is large for many i then the theoretical treatment becomes complicated. In this section, we make the order-estimations by imposing the following assumption.

Assumption F We assume the following for any i.

$$\|f_i(\boldsymbol{u})\| = O(\|u\|^{\nu+(n-i)\lambda}) \ \text{ so long as } \ \nu+(n-i)\lambda \ \text{ is an integer.} \tag{4.4}$$

Lemma 2 *For small $\|\boldsymbol{u}\|$ satisfying (4.4), we have the following order estimations, so long as $\boldsymbol{u}$ is not close to the zero-points of $f_n(\boldsymbol{u})$ and $\alpha_i(\boldsymbol{u})-\alpha_j(\boldsymbol{u})$ $(\forall i \neq j)$.*

$$\|\alpha_1\|,\ \|\alpha_1-\alpha_j\| \ = \ O(\|\boldsymbol{u}\|^{\lambda}) \quad (j=2,\ldots,n), \tag{4.5}$$

$$\|F'_{\mathrm{New}}(\alpha_i,\boldsymbol{u})\| \ = \ O(\|\boldsymbol{u}\|^{\nu+(n-1)\lambda}) \quad (i=1,\ldots,n), \tag{4.6}$$

$$\|F_h(\alpha_i,\boldsymbol{u})\| \ = \ o(\|\boldsymbol{u}\|^{\nu+n\lambda}) \quad (i=1,\ldots,n). \tag{4.7}$$

Proof Order estimations in (4.5) and (4.6) are direct consequences of Assumption A and Assumption F. Since each coefficient of $F_{\mathrm{h}}(x,\boldsymbol{u})$ is of higher order w.r.t. $u_1,\ldots,u_n$ than the corresponding coefficient of $F_{\mathrm{New}}(x,\boldsymbol{u})$ by at least 1, we obtain (4.7). ♢

Proposition 1 *Let $\delta\phi_1^{(k)}(\boldsymbol{u})$ be the coefficient of η^k-term of Hensel series $\phi_1^{(\infty)}(\boldsymbol{u},t)$. Then, except near the zero-points of $\alpha_i(\boldsymbol{u})-\alpha_j(\boldsymbol{u})$ $(\forall i \neq j)$, we have the following.*

$$\|\delta\phi_1^{(k)}(\boldsymbol{u})\| = o(\|\boldsymbol{u}\|^{\lambda+k-1}) \quad \text{for small} \ \ \|\boldsymbol{u}\| \ \ \text{satisfying} \ \ (4.4). \tag{4.8}$$

Proof Consider (3.4) with Lemma 2. Since $\delta F^{(1)} = F_{\mathrm{h}}(x,\boldsymbol{u})$, we see

$$\|\delta\phi_1^{(1)}(\boldsymbol{u})\| \ = \ \frac{O(\|F_{\mathrm{h}}(\alpha_1,\boldsymbol{u})\|)}{O(\|F'_{\mathrm{New}}(\alpha_1,\boldsymbol{u})\|)} \ = \ \frac{o(\|\boldsymbol{u}\|^{\nu+n\lambda})}{O(\|\boldsymbol{u}\|^{\nu+(n-1)\lambda})} = o(\|\boldsymbol{u}\|^{\lambda}).$$

Hence, (4.8) is valid for $k=1$. Suppose the proposition is valid up to $k-1$, hence we have

$$\frac{\|\delta F^{(j)}(\alpha_i,\boldsymbol{u})\|}{\|F'_{\mathrm{New}}(\alpha_i,\boldsymbol{u})\|} = o(\|\boldsymbol{u}\|^{\lambda+j-1}) \quad \text{for small} \ \ \|\boldsymbol{u}\|, \qquad j=1,\ldots,k-1.$$

Then, we can order-estimate $\|\delta F^{(k)}(\alpha_1,\boldsymbol{u})\|$ in (3.6), $k\geq 2$, as follows.

$$O\Big(\sum_{j=2}^{n} \frac{\|F'_{\mathrm{New}}(\alpha_1,\boldsymbol{u})\|}{\|\alpha_1-\alpha_j\|} \times \Big[\sum_{k'=1}^{k-1} \frac{\|\delta F^{(k')}(\alpha_1,\boldsymbol{u})\|}{\|F'_{\mathrm{New}}(\alpha_1,\boldsymbol{u})\|} \frac{\|\delta F^{(k-k')}(\alpha_j,\boldsymbol{u})\|}{\|F'_{\mathrm{New}}(\alpha_j,\boldsymbol{u})\|}\Big]\Big)$$

$$= \ \frac{O(\|\boldsymbol{u}\|^{\nu+(n-1)\lambda})}{O(\|\boldsymbol{u}\|^{\lambda})} \cdot \sum_{k'=1}^{k-1} o(\|\boldsymbol{u}\|^{\lambda+k'-1})\, o(\|\boldsymbol{u}\|^{\lambda+(k-k')-1}) \ = \ o(\|\boldsymbol{u}\|^{\nu+n\lambda+k-1}).$$

Therefore, $\|\delta\phi_1^{(k)}(\boldsymbol{u})\| = \|\delta F^{(k)}(\alpha_1,\boldsymbol{u})\|/\|F'_{\mathrm{New}}(\alpha_1,\boldsymbol{u})\| = o(\|\boldsymbol{u}\|^{\lambda+k-1})$. ♢

We remark that the order estimation in (4.8) is the same as that in monic case.

5 On convergence property near the expansion point

The formula (3.4) with (3.6) is very useful for analyzing the properties of Hensel series theoretically. In [16], we have done such an analysis in the case of monic defining polynomials. In this section, we perform a similar analysis in the nonmonic case. We will see that, except near the zero-points of the leading coefficient, the regularized Hensel series in the nonmonic case show quite similar convergence behaviors as those in the monic case.

Theorem 2 *In a neighborhood of the expansion point, except near the zero-points of $f_n(\boldsymbol{u})$, any divergence domain of regularized Hensel series $\bar{\phi}_1^{(\infty)}(\boldsymbol{u},1)$ starts from the expansion point and spreads outside radially along the zero-points of $\alpha_1(\boldsymbol{u})-\alpha_j(\boldsymbol{u})$, $2\leq j\leq n$.*

Proof $G_1^{(k)}(x, \boldsymbol{u}, t)$ in (3.4) with (3.6) tells us that, in the neighborhood of the expansion point, $\bar{\phi}_1^{(\infty)}(\boldsymbol{u}, 1)$ diverges on the zero-points of $F_{\mathrm{New}}(\alpha_1, \boldsymbol{u}) = f_n(\boldsymbol{u}) \prod_{j=2}^{n} (\alpha_1(\boldsymbol{u}) - \alpha_j(\boldsymbol{u}))$, may diverge on zero-points of $f_n(\boldsymbol{u})$, and on no other point. ♢

Theorem 3 *In the neighborhood of the expansion point, two branches $\bar{\varphi}_1(\boldsymbol{u})$ and $\bar{\varphi}_j(\boldsymbol{u})$ $(j \geq 2)$ of the regularized algebraic function cross along the zero-points of $\alpha_1(\boldsymbol{u}) - \alpha_j(\boldsymbol{u})$ if $\alpha_1(\boldsymbol{u})$ and $\alpha_j(\boldsymbol{u})$ cross, or they are tangent to each other if $\alpha_1(\boldsymbol{u})$ and $\alpha_j(\boldsymbol{u})$ are so.*

Proof Theorem 2 tells that, in the neighborhood of the expansion point, the divergence domains spread from the expansion point radially along the zero-points of $\alpha_1(\boldsymbol{u}) - \alpha_j(\boldsymbol{u})$, and Proposition 1 says that the Hensel series are well approximated by the initial terms. Furthermore, algebraic functions are also approximated well by the roots of the Newton polynomial. Therefore, near the zero-points of $\alpha_1(\boldsymbol{u}) - \alpha_j(\boldsymbol{u})$, $\bar{\varphi}_1(\boldsymbol{u})$ either crosses with $\bar{\varphi}_j(\boldsymbol{u})$ or is tangent to $\bar{\varphi}_j(\boldsymbol{u})$. ♢

Theorem 4 *Let S_r be the surface of the hypersphere, $\|\boldsymbol{u}\|^2 = r^2$, where r is a small real positive number. Suppose S_r contains zero-points of $f_n(\boldsymbol{u})$ on which $\alpha_1(\boldsymbol{u})$ diverges, and let δS_r be small neighborhood of the zero-points, on S_r. Let $\check{S}_r$ be $S_r - \delta S_r$. Then, we have*

$$\frac{[\text{divergence area of } \bar{\phi}_1(\boldsymbol{u}) \text{ on } \check{S}_r]}{[\text{convergence area of } \bar{\phi}_1(\boldsymbol{u}) \text{ on } \check{S}_r]} \to 0 \quad \text{as} \quad r \to 0. \tag{5.1}$$

Proof The regularized root $\bar{\alpha}_1(\boldsymbol{u})$ does not diverge on $\check{S}_r$, and the higher order regularized terms $\overline{\delta\phi}_1^{(k)}(\boldsymbol{u})$ $(k \geq 1)$ diverge only at the zero-points of $F_{\mathrm{New}}(\alpha_1, \boldsymbol{u})$. Lemma 2 tells us that $\|\delta\phi_1^{(k)}(\boldsymbol{u})\| / \|\alpha_1(\boldsymbol{u})\| = o(\|\boldsymbol{u}\|^{k-1})$ for small $\|\boldsymbol{u}\|$ except near the zero-points of $F_{\mathrm{New}}(\alpha_1, \boldsymbol{u})$. Hence, the ratio in (5.1) becomes smaller and smaller as $r \to 0$. ♢

Corollary 1 *The regularized Hensel series $\bar{\varphi}_1^{(\infty)}(\boldsymbol{u}, 1)$ converges in most area of the small neighborhood of the expansion point.*

Example 3 (convergence domain near the origin) Consider $F_2(x, u, v)$ in Example 2. We compare the regularized truncated Hensel series $\bar{\phi}_1^{(4)}(u, v)$ with $\bar{\varphi}_i(u, v)$ $(i = 1, 2, 3)$ in the real region $-a \leq u, v \leq a$. Since we do not know the correspondence between the Hensel series and the algebraic functions, we compute the quantity

$$d = \min\{\, |\bar{\phi}_1^{(4)}(u, v) - \bar{\varphi}_i(u, v)|,\ i = 1, 2, 3 \,\}$$

at many points in the region and draw a domain in which we have $d \leq \delta$ ($\delta = 0.01$ or $\delta = 0.001$). Figures 1a and 1b show the results; gray areas are "convergence domains".

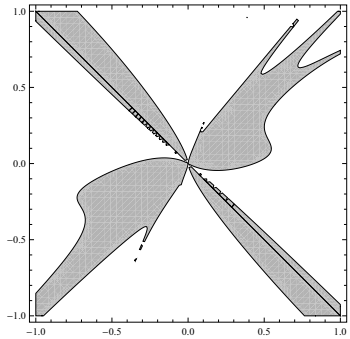


Fig. 1a ($a = 1.0$, $\delta = 0.01$)

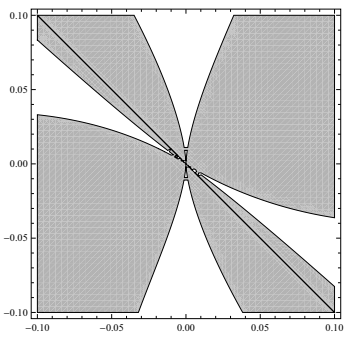


Fig. 1b ($a = 0.1$, $\delta = 0.001$)

We see from Fig. 1a that the "convergence domain" is pretty wide. On the other hand, looking around the origin, we notice that Theorem 4 seems to be unsatisfied. So, we investigate the "convergence domain" near the origin by increasing the plotting points. Figure 1b shows the result, which is consistent with Theorem 4.

The $\bar{\phi}_1^{(4)}(u,v)$ drawn in Figs. 1a and 1b corresponds to the initial factor $(u+v)x-1$. Hence, $\bar{\phi}_1^{(4)}(u,v)$ diverges on the line $u+v=0$. We see that the divergence domain near the line is very narrow, showing that the Hensel series truncated at $k=4$ approximates the corresponding algebraic function well even in a small neighborhood of the zero-points of the leading coefficient. On the other hand, we have $F'_{2\mathrm{New}}(\alpha_1,u,v)=u(2u+3v)/(u+v)$, hence $\bar{\phi}_1^{(4)}(u,v)$ diverges along lines $u=0$ and $2u+3v=0$. ♢

6 On many-valuedness of Hensel series

In [5], we observed that a Hensel series may jump from one branch of algebraic function to another when it passes a divergence domain, and we verified this property theoretically in [16] for monic defining polynomials. In this paper, we verify this property for nonmonic defining polynomials.

Definition 3 *If $\alpha_1(\boldsymbol{u}) \in \mathbb{C}[\boldsymbol{u}]$ then the Hensel series $\phi_1^{(\infty)}(\boldsymbol{u},1)$ is called* rational*, otherwise the series is called* algebraic*.*

Suppose $F_{\mathrm{New}}(x,\boldsymbol{u})$ is factored in $\mathbb{C}[x,\boldsymbol{u}]$ as $F_{\mathrm{New},1}(x,\boldsymbol{u})\cdots F_{\mathrm{New},r}(x,\boldsymbol{u})$, and let α_1 be a root of $F_{\mathrm{New},1}(x,\boldsymbol{u})$. If $\deg(F_{\mathrm{New},1})=m>1$ then there are m conjugate roots, let them be $\alpha_{1,1}(=\alpha_1),\ \alpha_{1,2},\ \ldots,\ \alpha_{1,m}$. Let the Hensel series $\phi_{1,i}^{(\infty)}$ correspond to $\alpha_{1,i}$ $(1\le i\le m)$. In [18], it was shown that truncated Hensel series $\phi_{1,1}^{(k)},\ldots,\phi_{1,m}^{(k)}$ are mutually conjugate hence $\phi_{1,1}^{(k)}(\boldsymbol{u})$ is m-valued.

Continuous singular points usually form a line when $\ell=2$, a surface when $\ell=3$, or an $(\ell$–$1)$-dimensional hypersurface when $\ell\ge 4$, and we call them *singularity lines* for simplicity.

Let P_0 and P_1 be points in $\mathbb{C}^\ell$, which are in a convergence domain of Hensel series $\phi_1^{(\infty)}(\boldsymbol{u},1)$, in the neighborhood of the expansion point. Let C_0 be a path which starts P_0, rounds a singularity line one time, and comes back to P_0. Let C_1 be a path which starts P_0 and arrives at P_1, without rounding any singularity line. The next theorem is obvious.

Theorem 5 *Tracing the truncated algebraic Hensel series $\phi_1^{(k)}(\boldsymbol{u},1)$ $(k\ge 2)$ along the path C_0, the series transfers to one of the conjugate Hensel series when it arrives at P_0.*

Theorem 6 *Tracing the truncated Hensel series $\bar{\phi}_1^{(k)}(\boldsymbol{u},1)$ $(k\ge 2)$ along the path C_1, we encounter the same Hensel series at P_1, no matter whether the series is rational or algebraic, and no matter whether the path passes a divergence domain or not.*

Proof Since the divergence domain spreads from the expansion point radially and the most area of the small neighborhood of the expansion point is the convergence domain, we can move the path C_1 in $\mathbb{C}^\ell$ so that it does not pass any divergence domain. Then, the value of $\bar{\phi}_1^{(k)}(\boldsymbol{u},1)$ changes continuously from P_0 to P_1, proving the theorem. ♢

Corollary 2 *Let $\bar{\phi}_1^{(\infty)}(\boldsymbol{u},1)$ correspond to a branch $\bar{\varphi}_1(\boldsymbol{u})$ of the original algebraic function. Tracing $\bar{\phi}_1^{(k)}(\boldsymbol{u},1)$ along C_1 by passing a divergence domain, it may jump to another branch of the original algebraic function.*

Proof Theorem 3 says that two branches of the algebraic function cross or tangent to each other in each divergence domain near the origin. Therefore, if we trace the Hensel series along C_1, the series either may stay on the same branch of the algebraic function or may jump to another branch, if these branches cross there. ♢

We show the jumping by a simple example.

Example 4 (jumping of Hensel series). Let $F_3(x,u,v)$ be as follows.

$$F_3(x,u,v) = [(u+v)x-1]\,(ux-1)\,(vx+1) + (u^4+v^4)\,x^2 + u^3v^3.$$

We trace truncated Hensel series $\phi_3^{(10)}(u,v)$ corresponding to the initial factor $vx+1$ and the algebraic functions $\varphi_i(u,v)$ $(i=1,2,3)$, along the circle C_3: $(u,v) = 0.1 \times (\cos\theta, \sin\theta)$ $(0 \le \theta \le 2\pi)$. (If we make the circle C_3 bigger, the graphs become more complicated.) We have $F'_{3\mathrm{New}}(-1/v,u,v) = (u+v)(u+2v)/v^2$, hence the divergence domain spreads along the lines $u+v=0$ and $u+2v=0$, and C_3 crosses the divergence domain at about $\theta = 3\pi/4,\ 5\pi/6,\ 7\pi/4,\ 11\pi/6$.

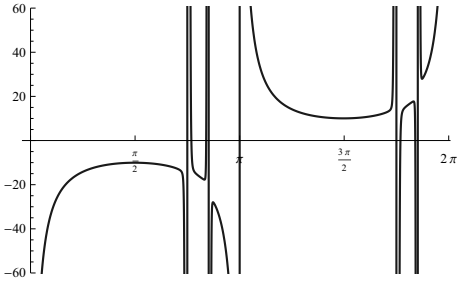


Fig. 2a $\phi_3^{(10)}(u,v)$

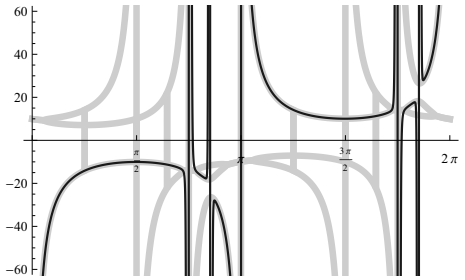


Fig. 2b $\varphi_i(u,v)$ $(i=1,2,3)$

In Fig. 2a, we show $\phi_3^{(10)}(u,v) = -1/v + \text{H.T.}$ traced along C_3; the graph diverges at $\theta = 0,\ 3\pi/4,\ 5\pi/6,\ \pi,\ 7\pi/4,\ 11\pi/6,\ 2\pi$. In Fig. 2b, we show real parts of $\varphi_i(u,v)$ $(i=1,2,3)$ traced along C_3 (gray curves); the value of φ_i becomes complex around $\theta = 0,\ 3\pi/4,\ \pi,\ 7\pi/4,\ 2\pi$. (We have used Mathematica to draw the figures in this paper. Mathematica draws only the principal values of algebraic functions, and four fake vertical lines in Fig. 2b are due to Mathematica.) For comparison, we show $\phi_3^{(10)}(u,v)$ traced along C_3 by black curves. We see that the Hensel series jumps from one branch to another when it passes $\theta = 5\pi/6,\ 11\pi/6$. Note that no jumping occurs at $\theta = 0,\ \pi,\ 2\pi$ where the leading coefficient vanishes. ♢

References and Notes

[1] E. L. Allgower and K. Georg, *Numerical Continuation Methods, an Introduction.* Springer-Verlag, 1990.

[2] F. Beringer and F. Jung, Multi-variate polynomials and Newton-Puiseux expansions. *Proc. SNSC 2001*, W. Winkler and U. Langer (Eds.): *Lec. Notes Comp. Sci.*, **2630**, 240–254, 2003.

[3] J. Della Dora and F. Richard-Jung, About the Newton algorithm for non-linear ordinary differential equation. *Proc. ISSAC'97*, W.W. Küchlin (Ed.), 298–304, ACM Press, 1997.

[4] D. Inaba, Factorization of multivariate polynomials by extended Hensel construction. *ACM SIGSAM Bulletin*, **39**(1), 2–14, 2005.

[5] D. Inaba and T. Sasaki, A numerical study of extended Hensel series. *Proc. SNC'07*, J. Verschede and S.T. Watt (Eds.), 103–109, ACM Press, 2007.

[6] M. Iwami, Analytic factorization of the multivariate polynomial. *Proc. CASC 2003*, V.G. Ganzha, E.W. Mayr and E.V. Vorozhtsov (Eds.), 213–225, Technishe Universität München Press, 2003.

[7] M. Iwami, Extension of expansion base algorithm to multivariate analytic factorization. *Proc. CASC 2004*, V.G. Ganzha, E.W. Mayr and E.V. Vorozhtsov (Eds.), 269–282, Technishe Universität München Press, 2004.

[8] J.D. Lambert, *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley and Sons, 1991.

[9] S. McCallum, On testing a bivariate polynomial for analytic reducibility. *J. Symb. Comput.*, **24**, 509–535, 1997.

[10] J. McDonald, Fiber polytopes and fractional power series. *J. Pure Appl. Algebra*, **104**, 213–233, 1995.

[11] P.D. González Pérez, Singularités quasi-ordinaires toriques et polyèdre de Newton du discriminant. *Canad. J. Math.*, **52**(2), 348–368, 2000.

[12] A. Poteaux, Computing monodromy groups defined by plane algebraic curves. *Proc. SNC'07*, J. Verschede and S.T. Watt (Eds.), 36–45, ACM Press, 2007.

[13] T. Sasaki and D. Inaba, Hensel construction of $F(x, u_1, \ldots, u_\ell)$, $\ell \geq 2$, at a singular point and its applications. *ACM SIGSAM Bulletin*, **34**(1), 9–17, 2000.

[14] T. Sasaki and D. Inaba, On series expansion of multivariate algebraic functions by extended Hensel construction. Preprint of Univ. Tsukuba, 2008. (submitted)

[15] T. Sasaki and D. Inaba, Multivariate Hensel construction in roots. Preprint of Univ. Tsukuba, 2008. (submitted)

[16] T. Sasaki and D. Inaba. Convergence and many-valuedness of Hensel series near the expansion point. *Proc. SNC 2009*, 159–167, ACM Press, Aug. 2009.

[17] T. Sasaki and F. Kako, Solving multivariate algebraic equation by Hensel construction. Preprint of Univ. Tsukuba, March, 1993.

[18] T. Sasaki and F. Kako, Solving multivariate algebraic equation by Hensel construction. *Japan J. Indust. Appl. Math.*, **16**(2), 257–285, 1999.

[19] T. Sasaki and S. Yamaguchi, An analysis of cancellation error in multivariate Hensel construction with floating-point number arithmetic. *Proc. ISSAC'98*, O. Gloor (Ed.), 1–8, ACM Press, 1998.

[20] K. Shiihara and T. Sasaki, Analytic continuation and Riemann surface determination of algebraic functions by computer. *Japan J. Indust. Appl. Math.*, **13**(1), 107–116, 1996.

[21] R.J. Walker. *Algebraic Curves*. Springer-Verlag, New York-Heidelberg-Berlin, 1950.

A Sequence of Nearest Polynomials with Given Factors

Hiroshi Sekigawa[1]

[1] NTT Communication Science Laboratories,
Nippon Telegraph and Telephone Corporation
3-1 Morinosato-Wakamiya, Atsugi-shi, Kanagawa, 243-0198 Japan
sekigawa@theory.brl.ntt.co.jp

Abstract

Let p and f_0 be given nonzero multivariate polynomials with real coefficients and $\|f_0\| = 1$, where $\|f_0\|$ is the Euclidean norm of f_0. For $j = 1, 2, \ldots$, let $p_{2j-1} = f_{j-1}g_j$ be the nearest polynomial to p such that f_{j-1} is a factor of p_{2j-1} and $\deg(p_{2j-1}) \leq \deg(p)$, where deg is the total degree, and $p_{2j} = c_j f_j g_j$ be the nearest polynomial to p such that g_j is a factor of p_{2j}, $\deg(p_{2j}) \leq \deg(p)$, $\|f_j\| = 1$, and $c_j > 0$. We investigate the behavior of the sequences $\{ p_j \}$, $\{ f_j \}$, $\{ g_j \}$, and $\{ c_j \}$.

1 Introduction

Since the mid 1990's, there have been many studies on the nearest polynomials to given polynomials with given properties. Finding the nearest polynomial with a given zero is a typical problem [4]. This problem can be generalized to finding the nearest polynomial with a given factor.

In this article, we measure the distance between two polynomials f and g by using $\|f - g\|$, the Euclidean norm of the vector of coefficients of $f - g$. Let p and f_0 be given nonzero multivariate polynomials with real coefficients with $\|f_0\| = 1$. For $j = 1, 2, \ldots$, let $p_{2j-1} = f_{j-1}g_j$ be the nearest polynomial to p such that f_{j-1} is a factor of p_{2j-1} and $\deg(p_{2j-1}) \leq \deg(p)$, where deg is the total degree, and $p_{2j} = c_j f_j g_j$ be the nearest polynomial to p such that g_j is a factor of p_{2j}, $\deg(p_{2j}) \leq \deg(p)$, $\|f_j\| = 1$, and $c_j > 0$. We consider the sequences $\{ p_j \}$, $\{ f_j \}$, $\{ g_j \}$, and $\{ c_j \}$. The following are natural questions. Do these sequences converge? If they converge to $\tilde{p}$, $\tilde{f}$, $\tilde{g}$, and $\tilde{c}$, respectively, does $\tilde{p} = \tilde{f}\tilde{g}$ or $\tilde{p} = \tilde{c}\tilde{f}\tilde{g}$ hold? The aim of this article is to answer these questions.

If f_j and g_{j+1} (or g_j) are not constant, $f_j g_{j+1}$ (or $c_j f_j g_j$) is an approximate factorization of p. Hence, by perturbing f_0, we might obtain a good approximate factorization of p.

2 Partial Answers to the Questions

Let P and F be nonzero polynomials in $\mathbb{R}[x_1, \ldots, x_s]$ with $\deg(P) = n$ and $\deg(F) = d$ ($1 \leq d < n$), V be $\{ H \in \mathbb{R}[x_1, \ldots, x_s] \mid \deg(H) \leq n \}$, and W be $\{ H \in V \mid F \text{ is a factor of } H \}$. V is a finite dimensional $\mathbb{R}$-vector space and W is an $\mathbb{R}$-subspace of V. Let Q be the orthogonal projection of $P \in V$ onto W. Then, Q is the unique nearest polynomial to P such that F is a factor of Q and $\deg(Q) \leq \deg(P)$.

Since $V \supset \{ H \in \mathbb{R}[x_1, \ldots, x_s] \mid \deg(H) \leq n - d \} \ni X \mapsto FX \in W$ is a linear map, we can represent the product of F as a matrix $M(F)$. Let $v(H)$ be the vector of coefficients

*Correspondence to: 3-1 Morinosato-Wakamiya, Atsugi-shi, Kanagawa, 243-0198 Japan. Tel +81 46 240 3644 (fax +81 46 240 4709).

of $H \in V$. Then, $P_1 = FG$ is the nearest polynomial if and only if $X = v(G)$ minimizes $\|v(P) - M(F)X\|$. $X = v(G)$ is the unique solution of the normal equation

$$M(F)^* M(F) X = M(F)^* v(P), \tag{1}$$

where $M(F)^*$ is the conjugate transpose of $M(F)$ (Theorem 2.1.2 in [1]). Therefore, we can easily obtain G. Note that $M(F)^* M(F)$ is nonsingular.

We define the sequences $\{ p_j \}$, $\{ f_j \}$, $\{ g_j \}$, and $\{ c_j \}$, as described in Section 1.

Proposition 1 *The following hold when $p_1 \neq 0$.*

1. *$\|p_j\|$ is monotone non-decreasing and converges.*
2. *$\lim_{j\to\infty} \|p_{j+1} - p_j\| = \lim_{j\to\infty} \|f_{j+1} - f_j\| = \lim_{j\to\infty} \|g_{j+1} - g_j\| = 0$ and $\lim_{j\to\infty} c_j = 1$.*

Remark 1 *g_1 and p_1 might be 0. For example, $g_1 = p_1 = 0$ when $s = 1$, $p = x_1^2 + x_1 + 1$, and $f_0 = (x_1 - 1)/\sqrt{2}$. If $g_1 \neq 0$, then p_j's, f_j's, and g_j's are nonzero for every j because $\|p_1\| > 0$ and $\|p_j\|$ is monotone non-decreasing.*

If $\|p_{N+1} - p\| = \|p_N - p\|$ holds for some N, then $p_j = p_N$ for every $j \geq N$. This follows from the fact that f_j (resp. g_j) is a factor of p_{2j} and p_{2j+1} (resp. p_{2j-1} and p_{2j}) and that p_{2j+1} (resp. p_{2j}) is the unique nearest polynomial to p having f_j (resp. g_j) as a factor.

To prove Proposition 1, we need the following lemma.

Lemma 1 *For f, $g \in \mathbb{R}[x_1, \ldots, x_s]$, there exist positive constants m and M depending on only $\deg(fg)$ such that $m \leq \frac{\|fg\|}{\|f\| \cdot \|g\|} \leq M$ hold.*

Proof. First we consider the case of $s = 1$. Let $fg = a_n x_1^n + \cdots + a_1 x_1 + a_0$ ($a_n \neq 0$). We can take 2^{-n} as m (Corollary 6.33 in [2]). Since the inequality $|a_j| \leq \|f\| \cdot \|g\|$ holds for every $0 \leq j \leq n$, we can take $\sqrt{n+1}$ as M.

Next we consider the case of $s \geq 2$. Since the Kronecker substitution $\sigma : \mathbb{R}[x_1, \ldots, x_s] \to \mathbb{R}[x]$ that maps x_j to $x^{n^{j-1}}$ ($n = \deg(fg)$) for $1 \leq j \leq s$ is a ring homomorphism and $\|h\| = \|\sigma(h)\|$ holds for any $h \in \mathbb{R}[x_1, \ldots, x_s]$, we can reduce this case to that of $s = 1$. ■

Proof of Proposition 1. Let $r_j = p - p_j$. $\|r_j\|$ converges because it is monotone non-increasing. Thus, $\|p_j\|$ is monotone non-decreasing and converges because $\|p_j\|^2 = \|p\|^2 - \|r_j\|^2$. Note that $\langle p_j, r_j \rangle = 0$, where $\langle \cdot, \cdot \rangle$ is the inner product of $\mathbb{R}^n$. Since $\|p_{j+1} - p_j\|^2 = \|r_j\|^2 - \|r_{j+1}\|^2$ holds, $\|p_{j+1} - p_j\|$ converges to 0. Note that $p_{j+1} - p_j = r_j - r_{j+1}$ and $\langle r_{j+1}, r_{j+1} - r_j \rangle = 0$.

From Lemma 1, there exist positive constants m and M not depending on j such that

$$\|p_{2j+2} - p_{2j+1}\| = \|g_{j+1}(c_{j+1} f_{j+1} - f_j)\| \geq m \|g_{j+1}\| \cdot \|c_{j+1} f_{j+1} - f_j\|,$$
$$\|p_1\| \leq \|p_{2j+1}\| = \|f_j g_{j+1}\| \leq M \|f_j\| \cdot \|g_{j+1}\| = M \|g_{j+1}\|.$$

Therefore, $\|c_{j+1} f_{j+1} - f_j\| \leq \frac{M \|p_{2j+2} - p_{2j+1}\|}{m \|p_1\|}$ holds and $\|c_{j+1} f_{j+1} - f_j\|$ converges to 0. The triangle inequality implies that

$$\|c_{j+1} f_{j+1} - f_j\| \geq | \, \|c_{j+1} f_{j+1}\| - \|f_j\| \, | = |c_{j+1} - 1|.$$

Thus, $\lim_{j\to\infty} c_j = 1$. This implies that $\|f_{j+1} - f_j\| \to 0$ because

$$\|f_{j+1} - f_j\| = \|f_{j+1} - c_{j+1} f_{j+1} + c_{j+1} f_{j+1} - f_j\| \leq |1 - c_{j+1}| + \|c_{j+1} f_{j+1} - f_j\|.$$

The equality $p_{2j+1} - p_{2j} = f_j(g_{j+1} - c_j g_j)$ implies that $\|g_{j+1} - c_j g_j\| \to 0$. Thus, $\|g_{j+1} - g_j\| \to 0$ because

$$\|g_{j+1} - g_j\| = \|g_{j+1} - c_j g_j + c_j g_j - g_j\| \leq \|g_{j+1} - c_j g_j\| + |c_j - 1| \cdot \|g_j\|$$

and $\{ \|g_j\| \}$ is bounded. ■

The sequences $\{ p_j \}$, $\{ f_j \}$, and $\{ g_j \}$ have limit points because they are bounded.

Theorem 1 *The following hold when $p_1 \neq 0$.*

1. *$\{ p_j \}$ converges or there are infinitely many limit points.*
2. *Let $\{ p_{j_k} \}$ be a convergent subsequence of $\{ p_j \}$.*
   (a) *$\{ f_{j_k} \}$ and $\{ g_{j_k} \}$ are convergent sequences and $\tilde{p} = \tilde{f}\tilde{g}$, where $\tilde{p} = \lim_{k\to\infty} p_{j_k}$, $\tilde{f} = \lim_{k\to\infty} f_{j_k}$, and $\tilde{g} = \lim_{k\to\infty} g_{j_k}$.*
   Furthermore, for the univariate case, if $\tilde{p} \neq p$, then $\deg(\gcd(\tilde{f}, \tilde{g})) \geq 1$ holds.
   (b) *$\tilde{p}$ satisfies the following two conditions.*
   - *$\tilde{p}$ is the unique nearest polynomial to p such that $\tilde{f}$ is a factor of $\tilde{p}$ and $\deg(\tilde{p}) \leq \deg(p)$.*
   - *$\tilde{p}$ is the unique nearest polynomial to p such that $\tilde{g}$ is a factor of $\tilde{p}$ and $\deg(\tilde{p}) \leq \deg(p)$.*

Proof. Statement 1 immediately follows from Lemma 2 below.

Let $\tilde{p}_k = p_{j_k}$, $\tilde{f}_k = f_{\lfloor j_k/2 \rfloor}$, and $\tilde{g}_k = g_{\lfloor (j_k+1)/2 \rfloor}$. Then, $\tilde{p}_k = \tilde{c}_k \tilde{f}_k \tilde{g}_k$ holds, where $\tilde{c}_k = 1$ or $c_{\lfloor (j_k+1)/2 \rfloor}$. Let $\{ \tilde{f}_{j_k} \}$ be a convergent subsequence of $\{ \tilde{f}_j \}$. We write $\tilde{p}_{j_k}$, $\tilde{f}_{j_k}$, $\tilde{g}_{j_k}$, and $\tilde{c}_{j_k}$ as $\hat{p}_k$, $\hat{f}_k$, $\hat{g}_k$, and $\hat{c}_k$, respectively. Let $\hat{f}$ be $\lim_{j\to\infty} \hat{f}_j$. Noting that $\hat{p}_j = \hat{c}_j \hat{f}_j \hat{g}_j$, we have $\tilde{p} = \hat{f}\hat{g}_j + (\hat{c}_j \hat{f}_j - \hat{f})\hat{g}_j + (\tilde{p} - \hat{p}_j)$. This equation implies that $\hat{f}$ is a factor of $\tilde{p}$, $\hat{g}_j$ converges, and $\tilde{p} = \hat{f} \cdot \lim_{j\to\infty} \hat{g}_j$ because $(\hat{c}_j \hat{f}_j - \hat{f})\hat{g}_j$ and $\tilde{p} - \hat{p}_j$ converge to 0. Since the number of polynomials h's that are factors of $\tilde{p}$ and $\|h\| = 1$ is finite, Lemma 2 implies that $\{ \tilde{f}_j \}$ is a convergent sequence. Similar arguments hold for $\{ \tilde{g}_j \}$.

For the univariate case, when $\tilde{p} \neq p$ let $\deg(\tilde{f}) = d$, $\deg(\tilde{g}) = e$, and

$$V = \{ h \in \mathbb{R}[x_1] \mid \deg(h) \leq d + e \},$$
$$V_1 = \{ \tilde{f}h \mid h \in \mathbb{R}[x_1], \deg(h) \leq e \}, \quad V_2 = \{ \tilde{g}h \mid h \in \mathbb{R}[x_1], \deg(h) \leq d \}.$$

V_1 and V_2 are $\mathbb{R}$-subspaces of the $\mathbb{R}$-vector space V. The dimensions of V, V_1, and V_2 are $d+e+1$, $e+1$, and $d+1$, respectively. For every $v_1 \in V_1$ and $v_2 \in V_2$, $\langle p - \tilde{p}, v_1 \rangle = \langle p - \tilde{p}, v_2 \rangle = 0$ hold. Therefore, $\langle p - \tilde{p}, w \rangle = 0$ for every $w \in W = V_1 + V_2$. Thus, $\dim(W) \leq d + e$ and $\dim(V_1 \cap V_2) \geq 2$ since $p - \tilde{p} \neq 0$. Take a nonzero polynomial $h \in (V_1 \cap V_2) \setminus \{ a\tilde{f}\tilde{g} \mid a \in \mathbb{R} \}$. If $\deg(h) = d + e$ we can take $c \in \mathbb{R}$ such that $\deg(h - c\tilde{f}\tilde{g}) < d + e$ because $\deg(\tilde{f}\tilde{g}) = d + e$. Since $0 \neq h - c\tilde{f}\tilde{g} \in (V_1 \cap V_2) \setminus \{ a\tilde{f}\tilde{g} \mid a \in \mathbb{R} \}$, by replacing h with $h - c\tilde{f}\tilde{g}$, if necessary, we can assume that $h \neq 0$ and $\deg(h) < d + e$. Therefore, $\deg(\gcd(\tilde{f}, \tilde{g})) \geq 1$ holds because $h \in V_1 \cap V_2$; that is, both $\tilde{f}$ and $\tilde{g}$ are factors of h.

The normal equation (1) implies that the nearest polynomial $\tilde{p}$ to p such that f is a factor of $\tilde{p}$ and $\deg(\tilde{p}) \leq \deg(p)$ is continuous with respect to f. If q ($\neq \tilde{p}$) is the nearest polynomial to p such that $\tilde{f}$ is a factor of q and $\deg(q) \leq \deg(p)$, $\|p - q\| < \|p - \tilde{p}\|$ holds. Then, $\|p - \tilde{p}_j\| < \|p - \tilde{p}\|$ holds for sufficiently large j. This contradicts the fact that $\|p - \tilde{p}_j\|$ is monotone non-increasing and converges to $\|p - \tilde{p}\|$. Similar arguments hold for $\tilde{g}$. ■

Lemma 2 *Let* $\{ a_j \} \subset \mathbb{R}^n$ *and* $a \in \mathbb{R}^n$. *When* $\|a_j - a\|$ *is monotone non-increasing and* $\lim_{j\to\infty} \|a_{j+1} - a_j\| = 0$, $\{ a_j \}$ *converges or there are infinitely many limit points.*

Proof. Let $\overline{B} = \{ c \in \mathbb{R}^n \mid \|c - a\| \leq \|a_1 - a\| \}$. Note that there exists at least one limit point of $\{ a_j \}$ because $\{ a_j \} \subset \overline{B}$ and $\overline{B}$ is compact.

It is sufficient to prove that if the number of limit points of $\{ a_j \}$ is finite, the number is one and $\{ a_j \}$ converges. Let $\tilde{a}_1, \ldots, \tilde{a}_t$ $(t < \infty)$ be all the limit points and $B_j(\delta) = \{ c \in \mathbb{R}^n \mid \|c - \tilde{a}_j\| < \delta \}$ $(\delta > 0)$. For any $\epsilon > 0$, we can take δ such that $0 < \delta < \epsilon$ and $B_j(2\delta) \cap B_k(2\delta) = \emptyset$ $(j \neq k)$. If there are infinitely many a_j's in $A(\delta) = B_1(\delta)^c \cap \cdots \cap B_t(\delta)^c$, where $B_k(\delta)^c = \mathbb{R}^n \setminus B_k(\delta)$, there exists a limit point in $A(\delta)$ because $A(\delta) \cap \overline{B}$ is compact and $\{ a_j \} \subset \overline{B}$. This contradicts the assumption that $\tilde{a}_1, \ldots, \tilde{a}_t$ are all the limit points. Thus, $\{ a_j \} \cap A(\delta)$ is a finite set. Hence, there exists $N \in \mathbb{N}$ such that $a_k \notin \{ a_j \} \cap A(\delta)$ and $\|a_{k+1} - a_k\| < \delta$ for every $k \geq N$. Let μ be the number such that $a_N \in B_\mu(\delta)$. Then, $a_k \in B_\mu(\delta)$ for every $k \geq N$; that is, $\|a_k - \tilde{a}_\mu\| < \delta < \epsilon$. Therefore, there exists only one limit point $\tilde{a}_\mu$ and $\lim_{k\to\infty} a_k = \tilde{a}_\mu$. ■

3 Conclusion

We investigated some properties of the sequences $\{ p_j \}$, $\{ f_j \}$, $\{ g_j \}$, and $\{ c_j \}$. One of the directions of future research is to study algorithms for finding an approximate factorization of a given polynomial based on the properties of the sequences. The algorithms must be compared with existing methods such as proposed in [3] and in the references there. Another direction is to prove that $\{ p_j \}$ always converges or to find an example $\{ p_j \}$ that does not converge.

Acknowledgements

This work was supported by the Japan Society for the Promotion of Science through a Grant-in-Aid for Scientific Research (KAKENHI) 21500026.

References and Notes

[1] S. L. Campbell and C. D. Meyer, Jr., *Generalized Inverses of Linear Transformations*, Pitman Publ. Ltd., London, 1979.

[2] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra, 2nd ed.*, Cambridge University Press, Cambridge, New York, Melbourne, Madrid, Cape Town, 2003.

[3] E. Kaltofen, J. P. May, Z. Yang, and L. Zhi, Approximate factorization of multivariate polynomials using singular value decomposition, *J. Symbolic Comput.*, Vol. 43, No. 5, pp. 359–376, 2008.

[4] H. J. Stetter, The nearest polynomial with a given zero, and similar problems, *ACM SIGSAM Bulletin*, Vol. 33, No. 4, pp. 2–4, 1999.

The Implementation and Complexity Analysis of the Branch Gröbner Bases Algorithm over Boolean Ring

Yao Sun[1], Dingkang Wang[2*]

Key Laboratory of Mathematics Mechanization,
Institute of Systems Science, AMSS, Academia Sinica
Beijing 100190, China
[1] sunyao@amss.ac.cn, [2] dwang@mmrc.iss.ac.cn

Abstract

We give an implementation and complexity analysis of an algorithm for computing branch Gröbner basis over the boolean ring. Our algorithm combines a modified F5 algorithm and strategies of branches for boolean polynomial system, and the implementation is based on ZDD data structure. The efficiency of the algorithm depends on the number of branches, which is determined by the branch-strategy. We give an upper bound on the number of branches in complexity analysis of the algorithm. In practice, the branch Gröbner bases algorithm performs very well for randomly generated systems as well as a class of stream cipher systems.

1 Introduction

Solving system of polynomial equations is a basic problem in computer algebra, through which many practical problems can be solved easily. Among all the methods for this purpose, Gröbner bases method, characteristic set method and resultant method are the most famous ones [8].

Since Buchberger proposed the Gröbner bases algorithm in 1965. This algorithm has been improved by many researchers, both from the data structure and the criteria to remove the redundant S-pairs. Now the most famous Gröbner bases algorithms are the F4 and F5 algorithms proposed by Faugère [3, 4]. The F4 algorithm import the matrix technique to make the reduction process more efficient, while the F5 algorithm presents two new criteria to eliminate the useless S-pairs, which can be definitely reduced to 0.

So far, both F4 and F5 algorithms have been implemented. The most efficient implementation of F4 algorithm is presented by Steel, and is available on the computer algebraic system Magma, while the the most efficient version of F5 algorithm is implemented by Faugère himself, which is not open. However, the F4 algorithm is still not perfect, as the high efficiency leads to the cost of enormous memories. For example, attacking the cryptographic system HFE80 by using F4 algorithm in Magma will cost nearly 16G memories.

For solving system of boolean polynomial equations, a Gröbner bases algorithm based on the ZDD data structure has been proposed by Brickenstein in 2007 (the PolyBoRi framework)[2]. His algorithm works very well for computing Gröbner basis with the pure lexicographic monomial order. However, since it is very expensive to compute the total degree leading monomial for a polynomial in the ZDD form, this algorithm possibly does not perform very well with total degree orders. So in our implementation, we prefer a new graded expression of polynomials such that the leading monomial for total degree order

*Correspondence to: Zhongguancun East Road 55, Beijing 100190, China. 0086-10-62541834.
Partially supported by NSFC 60821002/F02 and 10771206

can be calculated extremely fast, so our algorithm is more efficient with graded monomial orders. A characteristic set method for solving system of boolean polynomial equations is presented by Gao, and his method has pretty good performance on the problem of stream cipher systems [6]. Gao's implementation is also based on ZDD data structure and he use the branch technique to compute the ascending set series.

The success of Gao's algorithm is a motivation of our research on branch Gröbner basis. Our algorithm makes improvements both on saving the usage of memories and limiting the size of matrices. That is, on one hand, we utilize the ZDD data structure to save polynomials and decrease the cost of space, and on the other hand, we make new branches when the matrix grows bigger so that we only need to handle matrix with a reasonable size.

In theory, we employ a modified matrix F5 algorithm. More details can be found in [7]. In this paper, we will concentrate on the implementation and complexity analysis of our algorithm. The contents of this paper are organized as follow: the second section involves some preliminaries and the modified algorithm; the third section introduce the ZDD data structure and some sub-algorithms; complexity analysis comes in the forth section; some examples and timings are given in the fifth section; we will end this paper with conclusions.

2 The Algorithm

2.1 Notations

Let F_2 be the finite field with two elements 0 and 1, and $X = x_1, \cdots, x_n$ stands for the set of variables. Let H be the set of field polynomials $\{x_1^2 + x_1, \cdots, x_n^2 + x_n\}$, then the ring $\mathrm{R}_2 = \mathrm{F}_2[X]/ < H >$ is actually a boolean ring, where $< H >$ is the ideal generated by H in $\mathrm{F}_2[X]$, and in addition, we call the elements in R_2 boolean polynomials.

Let N be the set of non-negative integer and T be the power set of X, which means $T = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} | \alpha_i \in \{0, 1\}, i = 1, \cdots, n\}$. Assume $\prec$ is an admissible monomial order defined over T, then given $t = x_1^{\alpha_1} \cdots x_n^{\alpha_n} \in T$, we define the degree of t as $\deg(t) = \sum_{i=1}^{n} \alpha_i$. For a polynomial $0 \neq f \in \mathrm{F}_2[X]$, we have $f = \sum x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. Define the degree of f as $\deg(f) = \max\{\alpha_1 + \cdots + \alpha_n\}$, while the leading monomial of f is $\mathrm{lm}(f) = \max_{\prec}\{x_1^{\alpha_1} \cdots x_n^{\alpha_n}\}$.

2.2 Definitions

In order to introduce our algorithm, some definitions are necessary and the following definitions are imported from [7].

Definition 2.1 *Let $L = T \times \mathrm{F}_2[X] \times \mathrm{N} \times \mathrm{F}_2[X] \times \mathrm{N}$, and a labeled polynomial is a five-tuple vector $\mathcal{G} = (x^\alpha, f, i, g, k) \in L$. We define the signature of $\mathcal{G}$ as $\mathrm{S}(\mathcal{G}) = x^\alpha$, the initial as $\mathrm{init}(\mathcal{G}) = f$, the extended signature as $\mathrm{ES}(\mathcal{G}) = \mathrm{S}(\mathcal{G})\mathrm{lm}(\mathrm{init}(\mathcal{G})) = x^\alpha \mathrm{lm}(f)$, the index as $\mathrm{index}(\mathcal{G}) = i$, the polynomial as $\mathrm{poly}(\mathcal{G}) = g$ and the number as $\mathrm{num}(\mathcal{G}) = k$.*

Definition 2.2 *Let $\mathcal{F}, \mathcal{G} \in L$ be two labeled polynomials. We say $\mathcal{F} \prec_{es} \mathcal{G}$ (or $\mathcal{G} \succ_{es} \mathcal{F}$), if one of the following three cases is satisfied: 1. $\mathrm{ES}(\mathcal{F}) \prec \mathrm{ES}(\mathcal{G})$. 2. $\mathrm{ES}(\mathcal{F}) = \mathrm{ES}(\mathcal{G})$ and $\mathrm{index}(\mathcal{F}) > \mathrm{index}(\mathcal{G})$. 3. $\mathrm{ES}(\mathcal{F}) = \mathrm{ES}(\mathcal{G})$, $\mathrm{index}(\mathcal{F}) = \mathrm{index}(\mathcal{G})$ and $\mathrm{num}(\mathcal{F}) > \mathrm{num}(\mathcal{G})$.*

The order of labeled polynomials, which alleviate the influence of the order of the input polynomials or the initial polynomials, is modified from the F5 algorithm. Therefore, the criteria should be revised correspondingly. In the following two definitions, let $\mathcal{F} \in L$ be a labeled polynomial and $B \subset L$ be a set of labeled polynomials.

Definition 2.3 *We say $\mathcal{F}$ is normalized by B, if there do not exist a labeled polynomial $\mathcal{G} \in B$ and a monomial $v \in T$, such that* $\mathrm{S}(\mathcal{F}) = v\mathrm{lm}(\mathrm{poly}(\mathcal{G}))$, $\mathcal{F} \succ_{es} v\mathrm{lm}(\mathrm{init}(\mathcal{F}))\mathcal{G}$. *Particularly, in the boolean ring* R_2, *the condition* $\mathrm{lm}(\mathrm{init}(\mathcal{F})) \nmid \mathrm{S}(\mathcal{F})$ *should hold as well.*

Definition 2.4 *We say $\mathcal{F}$ can be rewritten by B, if there exists a labeled polynomial $\mathcal{G} \in B$ and a monomial $v \in T$, such that* $\mathrm{S}(\mathcal{F}) = \mathrm{S}(v\mathcal{G})$, $\mathrm{index}(\mathcal{F}) = \mathrm{index}(\mathcal{G})$ *and* $\mathrm{num}(\mathcal{F}) < \mathrm{num}(\mathcal{G})$.

2.3 Criteria

Now, we will give two new criteria modified from the F5 algorithm without proofs. A partial proof can be found in [7], and the complete one will come in a future paper.

1. **Syzygy criterion:** Given a critical pair: $\mathrm{s}(\mathcal{G}_1, \mathcal{G}_2) = (m, u_1, \mathcal{G}_1, u_2, \mathcal{G}_2)$, where $u_1, u_2 \in T$ and $\mathcal{G}_1, \mathcal{G}_2 \in B$. If either $u_1\mathcal{G}_1$ or $u_2\mathcal{G}_2$ is not normalized by B, then the critical pair $\mathrm{s}(\mathcal{G}_1, \mathcal{G}_2)$ can be discarded.
2. **Rewritten criterion:** Given a critical pair: $\mathrm{s}(\mathcal{G}_1, \mathcal{G}_2) = (m, u_1, \mathcal{G}_1, u_2, \mathcal{G}_2)$, where $u_1, u_2 \in T$ and $\mathcal{G}_1, \mathcal{G}_2 \in B$. If either $u_1\mathcal{G}_1$ or $u_2\mathcal{G}_2$ can be rewritten by B, then the critical pair $\mathrm{s}(\mathcal{G}_1, \mathcal{G}_2)$ can be discarded.

Our non-branch Gröbner bases algorithm is nothing else than a general Buchberger algorithm except replacing the polynomials with the labeled polynomials, adding two criteria and using matrix reduction. After revising the two criteria, our algorithm has less influence from the input order of the initial polynomials than the F5 algorithm. Furthermore, the two revised criteria can also eliminate almost all the useless critical pairs as the F5 algorithm does. For semi regular systems, there does not exist the critical pairs that can be reduced to 0, too. In fact, we proved in [7] that the modifications in the comparison of two labeled polynomials do not affect the function of the criteria. To illustrate how the two new criteria work, we have tested some randomly generated boolean polynomial systems in section 4.

2.4 Trick

Although the two modified criteria can remove almost all the useless critical pairs generated during the computation, the total efficiency is not so good as we wished. One possible reason may be the conflict between the signature and the polynomial.

In fact, the motivation of the signature is to record the origin of the present label polynomial. Take a labeled polynomial, say $\mathcal{G} = (x^\alpha, f, i, g, k) \in L$, for example. The signature tells us that the polynomial g is obtained by reducing the polynomial $x^\alpha f_i$ with 'smaller' labeled polynomials under the order $\prec_{es}$. So the signature actually works as a clue of the computation, and that is exactly why the two criteria work. However, there exists a natural conflict between the signature and the polynomial. That is, during the computation, the label polynomials generated later sometimes have smaller size but with bigger signature. By our algorithm as well as the F5 algorithm, the label polynomial with a bigger signature should be dealt with later. It is possible that some polynomials of smaller size can not be used immediately and this may lead to more computations. One trick can be used to solve this conflict. The key idea is to clear the signatures of some simple polynomials and to append them to the initial polynomials. We have

Proposition 2.5 *Adding an initial polynomial with the largest index at any time will not affect the correctness of the two criteria.*

Although adding new polynomials will not affect the correctness of the two criteria, in order to keep the algorithm correct, we must add polynomials that are in the original ideal. Usually, we add new initial polynomials in the following cases:

1. The new generated polynomial has a very low total degree, such as 1 or 2.
2. The leading monomials of some present initial polynomials can be reduced by the new generated polynomial.

However, adding too many polynomials cannot speed up the algorithm. Because adding new initial polynomials is actually to cut off the relationship between this polynomial and the initial polynomials, which will weaken the criteria, since the system of initial polynomials may not be semi regular any more, and many useless pairs cannot be detected.

2.5 Branch Strategy

In both the F4 and F5 algorithms, the sizes of matrices in the computation grow quickly with the degree of critical pairs. Huge matrices occupies enormous memories and is difficult to deal with. In consideration of the complexity, a natural idea comes to us, and that is, we should prevent the degree of critical pairs growing too high.

In order to control the size of the matrix, we can add polynomials that are not in the ideal, but this will apparently make the output incorrect. Fortunately, the cases are better in the boolean ring R_2. Since in R_2, any boolean polynomial have only two possible values 0 and 1, which makes the branch algorithm available. We can clone the present system and add a new polynomial with the value 0 and 1 to them respectively such that two polynomial sets are obtained. The original ideal is the intersection of the ideals generated by these two polynomial sets. This fact is very important for solving the original system. Furthermore, after adding the new polynomial, each system may have smaller matrices and it is easier to be dealt with. This is the motivation of our branch algorithm.

Theoretically, any polynomials can be added. In consideration of the complexity, we usually add polynomials which are simple enough or which can be used to reduce other polynomials. When should we add polynomials? Due to our experiments, we prefer to adding polynomials when the degree of critical pairs is high enough. We can set a degree bound $D \in \mathrm{N}$. When the remaining pairs have higher degrees than D, we add a new polynomial, or equivalently we make a new branch.

We have many options to choose the new polynomials and it is very difficult to tell which is the best. So we list some alternatives that have good performance in the experiments.

1. The polynomial with degree one.
2. The polynomial which is a highest homogeneous part of some present polynomial.
3. The polynomial with a high reference degree, which is a parameter comes from the shared ZDD data structure.

After all, we can present our branch Gröbner bases algorithm.

In this algorithm, the step 2.3.4.1 is to clear the signature of a polynomial, and the step 2.3.5 is to introduce new polynomials so as to add new branches to the algorithm. In order to make the algorithm more efficient, some auxiliary data can be kept in $BranchSet$ as well, such as the number $index$, k, the set CP and so on.

Algorithm 1 — Branch Grönber bases algorithm

Input: An ordered polynomials set $F = (f_1, \cdots, f_m) \subset \mathrm{F}_2[X]$.

Output: The branch Gröbner bases $BranchGB$ of the ideal generated by $F \cup H$, where $H = \{x_1^2 + x_1, \cdots, x_n^2 + x_n\} \subset \mathrm{F}_2[X]$ are the field polynomials.

1 Set $\mathcal{F}_i := (1, f_i, i, f_i, i), i = 1, \cdots, m$, $\mathcal{F}_{m+i} := (1, x_i^2+x_i, m+i, x_i^2+x_i, m+i), i = 1, \cdots, n$.
$index := m + n$, $k := m + n$, $BranchSet := \{\{\mathcal{F}_i | i = 1, \cdots, k\}\}$, $BranchGB := \{\}$.
2 While $BranchSet \neq \emptyset$ do
2.1 Select a $B \in BranchSet$ and $BranchSet := BranchSet \setminus \{B\}$.
2.2 Generate $CP := \{\mathrm{s}(\mathcal{P}, \mathcal{Q}) | \mathcal{P}, \mathcal{Q} \in B\}$.
2.3 While $CP \neq \emptyset$ do
2.3.1 $d := \min\{\deg(c) | c \in CP\}$, $D := \{c \in CP | \deg(c) = d\}$ and $CP := CP \setminus D$.
2.3.2 $D' := \{c \in D | c$ is not satisfied either of the Syzygy or Rewritten criterion $\}$.
2.3.3 Reduce D' by matrix and collect the new generated labeled polynomials as F^+.
2.3.4 For $\mathcal{P} \in F^+$ do
2.3.4.1 If $\mathcal{P}$ is simple enough
then $index := index + 1$, $\mathcal{P} := (1, \mathrm{poly}(\mathcal{P}), index, \mathrm{poly}(\mathcal{P}), k + 1)$.
else $\mathrm{num}(\mathcal{P}) := k + 1$.
2.3.4.2 $k := k + 1$, $CP := CP \cup \{\mathrm{s}(\mathcal{P}, \mathcal{Q}) | \mathcal{Q} \in B\}$. $B := B \cup \{\mathcal{P}\}$.
2.3.5 If the minimal degree of CP is high enough, then
2.3.5.1 choose a new polynomial $p \in \mathrm{F}_2[X]$.
2.3.5.2 $index := index + 1$, $k := k + 1$
2.3.5.3 Set $\mathcal{P}' := (1, p+1, index, p+1, k)$, $BranchSet := BranchSet \cup \{B \cup \{\mathcal{P}'\}\}$.
2.3.5.4 Set $\mathcal{P} := (1, p, index, p, k)$, $CP := CP \cup \{\mathrm{s}(\mathcal{P}, \mathcal{Q}) | \mathcal{Q} \in B\}$, $B := B \cup \{\mathcal{P}\}$.
2.4 $BranchGB := BranchGB \cup \{\{\mathrm{poly}(\mathcal{P}) | \mathcal{P} \in B\}\}$.
3 Return $BranchGB$.

3 Complexity Analysis

3.1 The Principle for Making Branch

One of the motivations of our branch Gröbner bases algorithm is to decrease the complexity in each branch so as to lessen the total complexity for solving the boolean polynomial system. Therefore, if we hope our branch algorithm have better efficiency than the non-branch algorithms, we should constrain the branch number within a reasonable bound.

The complexity of Gröbner bases using the $F4$ and $F5$ algorithm is determined by a degree D_{reg}, which is the upper bound of the highest degree of matrices, say D_p, constructed during the computation. Then the total complexity of $F4$ or $F5$ algorithm is roughly $\mathrm{O}(n^{\omega D_{reg}})$, where n is the number of variables and ω is the efficiency of matrix elimination who has a bound $2 \leq \omega \leq 3$. Similarly, since we make a new branch when the degree of pairs exceed a degree D, the complexity of each branch is roughly $\mathrm{O}(n^{\omega' D})$, where ω' is efficiency of our matrix reduction. Assuming the number of branches is M, then the total complexity of our branch algorithm is $\mathrm{O}(Mn^{\omega' D})$. If we hope our branch algorithm perform better, the complexity $\mathrm{O}(Mn^{\omega' D})$ must smaller than $\mathrm{O}(n^{\omega D_{reg}})$. Roughly speaking, the number M should satisfy the following inequality:

$$Mn^{\omega' D} < n^{\omega D_{reg}}, \text{ or } M < n^{\omega D_{reg} - \omega' D}.$$

However, it is very difficult to predict the number of branches produced in our algorithm. What we can do is to set up a bound such that when the number of branches exceed it, we stop the program. Fortunately, the number of branches for one kind of examples is usually stable, so we can anticipate the general performance for all problems of this kind by induction. The principle above set up a criterion to check whether it is possible for our algorithm to have a good performance.

3.2 The Estimation of D_{reg}

Before the system is computed by the $F5$ algorithm, we cannot obtain the practical value of D_p, so we have to try to estimate it or give an upper bound for it. Fortunately, the upper bound D_{reg} for F5 algorithm can be obtained from [1]. For a general semi regular system in the boolean ring, D_{reg} can be achieved from the following series:

$$S_{m,n}(z) = \sum_{d\geq 0} h_{d,m}(n) z^d = (1+z)^n / \prod_{k=1}^{m} (1+z^{d_k}).$$

where m is the number of the initial polynomials $(f_1, \cdots, f_m)$ and $d_k = \deg(f_k)$. Then D_{reg} is the first d such that $h_{d,m}$ is non-positive. Therefore, the upper bound can be calculated easily when m, n and the d_ks are given.

3.3 Theoretical Analysis of Randomly Generated Systems

Faugère has claimed that almost all the systems are semi-regular systems when n is not too small. So the randomly generated systems can be supposed to be semi-regular systems naturally, and the estimation of D_{reg} is a powerful tool to analyze the complexity of the F5 algorithm as well as our branch algorithm.

For convenience, we only consider some special cases and the others can be analyzed in a similar way. We assume the initial polynomials are m quadratic polynomials in R_2 and n is the number of variables. So when $m = n$, we can draw a picture of D_{reg}.

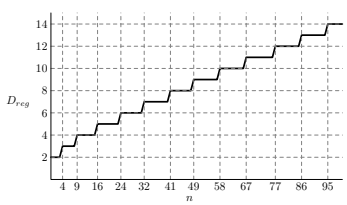


Fig.1 D_{reg} for $m = n$

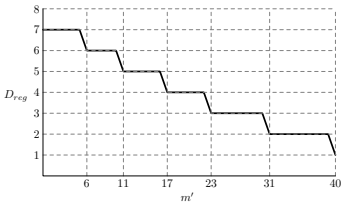


Fig.2 Degree 1

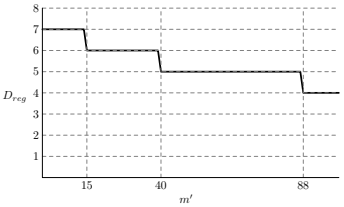


Fig.3 Degree 2

Although D_{reg} is only an upper bound of the practical highest degree D_p, the degree D_{reg} is reached almost all the time for randomly generated systems.

The motivation of our branch algorithm is to add new polynomials in order to lower the degree of matrices in the computation. Since randomly generated systems are semi-regular, so we can use the series in the last subsection to compute D_{reg} and then find out how many polynomials should be added such that D_{reg} becomes lower. If the number of polynomials is fixed, then the number of branches can be obtained easily.

For example, we consider the systems when $m = n = 40$ and in Fig.1, the corresponding D_{reg} is 7. By the serie, we can calculate the specific number of polynomials we should add

in order to lower the degree. Since adding different polynomials lead to different results, here we only consider two cases, one is adding polynomials with degree 1 and the other with degree 2. The Fig.2 shows how the D_{reg} varies with m', which is the number of new added polynomials. We can see that adding 6 polynomials of degree 1 will lower D_{reg} to 6, while adding 11 polynomials, the D_{reg} becomes 5 and so on. Then we obtain the following table. TH-Num is the theoretic branch number calculated by the inequality in subsection 3.1 with $\omega = \omega' = 2$. NumPoly is the smallest number of m' to lower D_{reg} to the corresponding degree and EXP-Num is the expected branch number calculated by NumPoly.

Table 1: Adding polynomials with degree 1

D_{reg}	6	5	4	3	2	1
TH-Num	1600	40^4	40^6	40^8	40^{10}	40^{12}
NumPoly	6	11	17	23	31	40
EXP-Num	64	2048	2^{17}	2^{23}	2^{31}	2^{40}
TH/EXP($\approx$)	$2^{4.6}$	$2^{10.3}$	$2^{14.9}$	$2^{19.6}$	$2^{22.2}$	$2^{23.9}$

In table 1, all the expected branch number are much smaller than the theoretical bound, so as we discussed in 3.1, our branch algorithm will have a better performance than the non-branch algorithm and the practical results in section 4 proves this. Remark that, the proportion in table 1 shows that adding 40 polynomials will lead to best efficiency, however, this is not the truth, since the matrices with lower degree are much denser than the bigger ones, then the parameter ω' is no longer 2 and combined with other factors, TH-Num will be much smaller than that in the table.

When adding polynomials with degree 2, the cases are not so good as that with degree 1, and Fig.3 shows the variation of D_{reg}. At least 15 quadratic polynomials should be added to lower D_{reg} to 6, while in Fig.2, only 6 is enough, so the corresponding expected branch number will be bigger than that in table 1. We can generate a similar table.

Table 2: Adding polynomials with degree 2

D_{reg}	6	5	4	3	2
TH-Num	1600	40^4	40^6	40^8	40^{10}
NumPoly	15	40	88	207	740
EXP-Num	2^{15}	2^{40}	2^{88}	2^{207}	2^{704}

The expected branch number are all bigger than the theoretic bound, so we can not expect a better performance of our branch algorithm. However, the practical cases are not so worse as the table shows, since the data in the table are calculated under a hypothesis that the new systems are still semi-regular systems, and so if we add some special polynomials which can reduce the present system significantly, the practical branch number can still be control in a reasonable bound.

Other examples can be analyzed in the same way as well, due to our experimental results, we can conclude that for almost all randomly generated systems, our branch algorithm has a better performance than the non-branch algorithms, such as F4 in Magma.

4 Computational Examples

In this section, some timings are imported from [7], but the complexity information are obtained by the method discussed in section 3. The branch strategy we used in the experiments is the first strategy in subsection 2.5 and all the timings are obtained from a computer (OS Linux, CPU Xion 4*3.0GHz, 16.0GB RAM).

4.1 The Criteria

In this subsection, we will see how the two modified criteria work. The initial polynomials are all randomly generated quadratic polynomials with $m = n$.

Table 3: The Criteria

n	6	8	10	12	14	16	18
T-pairs	198	740	2483	3296	94077	144801	211385
D-pairs	7	64	727	1146	46221	80308	112925
0-Polys	0	0	0	0	0	0	0
D/T(%)	3.54	8.65	29.28	34.77	49.13	55.46	53.42

In table 3, T-pairs stands for the total number of the pairs generated in the computation, while D-pairs is the number of the pairs detected by the two criteria and D/T is the proportion of these two number. Besides, 0-Polys is the number of polynomials that are reduced to 0 in the algorithm. From the table, all useless pairs are detected and the two criteria play an important role for improving the efficiency during the computation.

4.2 The Randomly Generated Systems

In this subsection we will see how the branch algorithm performs for randomly generated systems. In the following table, three groups of data will be presented. The first group involves the theoretical degree bound D_{reg} and the theoretical upper bound for the branch number M; The second group consists of the practical degree D_p computed by the F4 algorithm and the corresponding theoretical upper bound for M; in the last group, the degree D and the practical number of the branches is given. For the second and third group, the practical timings are given as well.

The initial polynomials are randomly generated quadratic polynomials with $m = n$. The practical degree D_p in the second group are obtained by the F4 algorithm in Magma. The upper bound of M is estimated with $\omega = \omega' = 2$. In the table 4, we use EST to represent the estimated data and MGB is the experiments data from the F4 algorithm, while BGB is that from our branch algorithm. TH- is short for theoretical and P- for practical and "-" means Magma runs out of memory.

Remark that in the table 4, the theoretical degrees for $n = 18, 20, 22$ are lower than the corresponding practical degrees, and the reason is that the F4 algorithm does not have so powerful criteria to remove all the useless pairs, so some useless computations are still done in the F4 algorithm. From the table, we can also see that the practical numbers of branches in our algorithm is always kept within the two theoretical bounds, so our branch algorithm will have a better performance and the timings prove that.

Table 4: The Randomly Generated Systems

	EST		*MGB*			*BGB*		
n	TH-Deg	TH-Num	P-Deg	TH-Num	Time	P-Deg	P-Num	Time
18	5	18^4	6	18^6	3.890	3	2048	0.791
20	5	20^4	6	20^6	14.220	3	8160	2.992
22	5	22^4	6	22^6	82.790	3	29046	13.235
24	6	24^6	–	–	–	3	65400	47.682
26	6	26^6	–	–	–	3	262018	149.121

4.3 The Stream Ciphers

In this section, we will use our branch algorithm to attack a class of stream ciphers, which is an important class of encryption algorithm. Here we only consider stream ciphers based on the linear feedback shift register (LFSR), and the filter functions are from [5].

- CanFil 2, $x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_4 + x_2x_5 + x_3 + x_4 + x_5$
- CanFil 3, $x_2x_3x_4x_5 + x_1x_2x_3 + x_2x_4 + x_3x_5 + x_4 + x_5$
- CanFil 8, $x_1x_2x_3 + x_2x_3x_6 + x_1x_2 + x_3x_4 + x_5x_6 + x_4 + x_5$
- CanFil 9, $x_2x_4x_5x_7 + x_2x_5x_6x_7 + x_3x_4x_6x_7 + x_1x_2x_4x_7 + x_1x_3x_4x_7 + x_1x_3x_6x_7 + x_1x_4x_5x_7 + x_1x_2x_5x_7 + x_1x_2x_6x_7 + x_1x_4x_6x_7 + x_3x_4x_5x_7 + x_2x_4x_6x_7 + x_3x_5x_6x_7 + x_1x_3x_5x_7 + x_1x_2x_3x_7 + x_3x_4x_5 + x_3x_4x_7 + x_3x_6x_7 + x_5x_6x_7 + x_2x_6x_7 + x_1x_4x_6 + x_1x_5x_7 + x_2x_4x_5 + x_2x_3x_7 + x_1x_2x_7 + x_1x_4x_5 + x_6x_7 + x_4x_6 + x_4x_7 + x_5x_7 + x_2x_5 + x_3x_4 + x_3x_5 + x_1x_4 + x_2x_7 + x_6 + x_5 + x_2 + x_1$
- CanFil 10, $x_1x_2x_3 + x_2x_3x_4 + x_2x_3x_5 + x_6x_7 + x_3 + x_2 + x_1$

The data here consists of two parts: one part involves the practical degree D_p, the theoretical upper bound for M and the practical time costed by the F4 algorithm in Magma; the other part includes the degree D which is given by the user, the practical branch number and the time used by our branch algorithm. Again, $\omega = \omega' = 2$ and MGB and BGB represent the F4 algorithm in Magma and our branch algorithm respectively.

Table 5: The Stream Ciphers

Filters	n	81			100			128		
		Time	Deg	Num	Time	Deg	Num	Time	Deg	Num
	MGB	18.730	7	81^8	32.930	7	100^8	–	–	–
CanFil2	*BGB*	0.027	3	9	0.172	3	66	0.357	3	40
	MGB	–	–	–	1.360	7	100^8	–	–	–
CanFil3	*BGB*	0.085	3	11	0.150	3	19	1.210	3	17
	MGB	49.460	7	81^8	12.590	7	100^8	–	–	–
CanFil8	*BGB*	0.046	3	27	0.170	3	190	0.371	3	140
	MGB	–	–	–	–	–	–	–	–	–
CanFil9	*BGB*	0.418	4	40	1.230	4	217	20.901	4	77
	MGB	331.880	7	81^8	–	–	–	–	–	–
CanFil10	*BGB*	0.131	3	99	0.612	3	501	1.296	3	340

Table 5 shows that the practical number of branches is much smaller than the estimated

upper bound, so it is not strange to see that our branch algorithm is more efficient than the F4 algorithm for this kind of problems.

5 Conclusion

In this paper, we present an implementation of the branch Gröbner bases algorithm and analyze the complexity briefly. The experimental results show that for both randomly generated systems and a class of Stream Ciphers problems, our branch algorithm has better efficiency than the F4 algorithm in Magma. However, there are some problems which are still not solved completely, for example, how to find a general strategy for all problems and how to anticipate the number of branches before the computation, and these can be investigated in the future.

6 Acknowledgements

We would like to thank Professor Gao Xiaoshan for his useful suggestions and Huang Zhenyu for the discussing of programming codes.

References and Notes

[1] Bardet, M., Faugère, J.C., Salvy, B.: Complexity of Grönber basis computation for Semi-regular Overdetermined sequences over F_2 with solutions in F_2. *In Proc. ICPPSS International Conference on Polynomial System Solving Paris, November 24-25-26 2004 in honor of Daniel Lazard* (2004)

[2] Brickenstein, M., Dreyer, A.: PolyBoRi: A Framework for Gröbner Basis Computations with Boolean Polynomials. *MEGA 2007, Austria* (2007)

[3] Faugère, J.C.: A new efficient algorithm for computing gröbner bases (f4). *Journal of Pure and Applied Algebra*, 139(1), 61-88 (1999)

[4] Faugère, J.C.: A new efficient algorithm for computing Grönber bases without reduction to zero (F5). *Symbolic and Algebraic Computation, Porc. Conferenz ISSAC 2002*, 75-83 (2002).

[5] Faugère, J.C., Ars, G.: An Algebraic Cryptanalysis of Nonlinear Filter Generators Using Gröbner Bases. *Reserch report 4739, Institut National de Recherche en Informatique et en Automatique, Lorraine* (2003)

[6] Gao, X.S., Chai, F.J., Yuan, C.M.: A Characteristic Set Method for Equation Solving in F2 and Applications in Cryptanalysis of Stream Ciphers. *Journal of Systems Science and Complexity*, 21, 191-208 (2008)

[7] Sun, Y., Wang, D.K.: Branch Gröbner Bases Algorithm over Boolean Ring (Chinese). Preprint (2009)

[8] Wu, W.T.: Basic Principles of Mechanical Theorem-proving in Elementary Geometries, *J. Sys. Sci. & Math. Scis.*, 4, 207-235 (1984). *J. Automated Reasoning*, 2, 221-252 (1986)

Computing Boolean Gröbner Bases within Linear Algebra

Akira Suzuki

Kobe University, RokkodaiCho 1-1, Kobe, Hyogo, 657-8501, Japan
sakira@kobe-u.ac.jp

1 Introduction

In these years, many application of boolean Gröbner bases have been studied. Thus more effective algorithms to compute them are demanded. For a such purpose, in this article, we propose a way to compute within linear algebra.

Computation of Gröbner bases by use of linear algebra has been studied [1, 2, 6], and an explicit algorithm to compute Gröbner bases within linear algebra is shown at [13]. So we can calculate boolean Gröbner bases by modifying them.

On the other hand, these methods have a problem of wasting computer memory. In this article, we show a method to compute boolean Gröbner bases which does not require additional memory allocation during the computation.

2 Boolean Gröbner Bases

In this section, we give several definitions and notations of boolean polynomial rings and boolean Gröbner bases, and give a brief view of their properties.

Definition 1 *A ring B with an identity is called* a boolean ring *if every element b of B is idempotent, i.e., $b^2 = b$.*

We can easily check that a boolean ring B is commutative and has characteristic 2. Furthermore, defining $a \vee b = a+b+a \cdot b$, $a \wedge b = a \cdot b$, and $\neg a = 1+a$, we see that $(B, \vee, \wedge, \neg)$ forms a boolean algebra. For further properties and examples, the reader should refer [3, 4, 5, 7, 8, 10]. In the rest of this article, let B be a boolean ring and $\bar{X} = \{X_1, \ldots, X_n\}$ be a set of distinct variables.

Definition 2 *The quotient ring $B[X_1, \ldots, X_n]/\langle X_1^2 + X_1, \ldots, X_n^2 + X_n \rangle$ become a boolean ring again, and we call it* a boolean polynomial ring *and denote it $B(X_1, \ldots, X_n)$. We call an element of $B(X_1, \ldots, X_n)$* a boolean polynomial.

Next we give a glance over the definition and properties of boolean Gröbner bases. For the detailed argument, the reader should refer [9, 11, 12, 14].

A power product $X_1^{l_1} \cdots X_n^{l_n}$ is called *a boolean power product* if each l_i is either 0 or 1, and we let $BT(\bar{X})$ be the set of the power products. We fix a term order $\prec$ on $T(\bar{X})$. Then it can be considered as an well-order on $BT(\bar{X})$ since $BT(\bar{X}) \subseteq T(\bar{X})$. Now, a boolean polynomial $f(\bar{X}) \in B(\bar{X})$ is uniquely represented by $b_l t_l + \cdots + b_1 t_1$ where $b_1, \ldots, b_l \in B \setminus \{0\}$, $t_1, \ldots, t_l \in BT(\bar{X})$, and $t_1 \prec \cdots \prec t_l$. Then, for $f(\bar{X})$, we can naturally define the leading term $lt(f) = t_l$, the leading coefficient $lc(f) = b_l$, and the leading monomial $lm(f) = b_l t_l$. For $F \subseteq B(\bar{X})$, we use the same notation $lt(F) = \{lt(f) \ : \ f \in F\}$ and $lm(F) = \{lm(f) \ : \ f \in F\}$. Next we define boolean-closedness and monomial reductions of boolean polynomials.

Definition 3 *A boolean polynomial $f \in B(\bar{X})$ is* boolean-closed *if $lc(f) \cdot f = f$.* The boolean closure *$bc(f)$ of f is defined by $bc(f) = lc(f) \cdot f$. Then we notice that $bc(f)$ is boolean-closed.*

For $F \subseteq BT(\bar{X})$, we let *the boolean closure* $bc(F) \subseteq BT(\bar{X})$ of F such that $\langle bc(F) \rangle = \langle F \rangle$ and that every element of $bc(F)$ is boolean-closed. We should note that $bc(F)$ is not uniquely determined from F, besides an easy algorithm to calculate $bc(F)$ from F exists.

Definition 4 *For a boolean polynomial $f \in B(\bar{X})$, let $a = lc(f)$, $t = lt(f)$ and $h = f + at$. Then,* a monomial reduction *$\rightarrow_f$ by f is defined by*

$$bts + p \rightarrow_f (1 + a)bts + absh + p$$

where $b \in B$ with $ab \neq 0$, $s \in BT(\bar{X})$ with $st \in BT(\bar{X})$, and $p \in B(\bar{X})$ which the term ts does not appear in p. For a set $F \subseteq B(\bar{X})$, we write $g \rightarrow_F g'$ if $g \rightarrow_f g'$ for some $f \in F$. A recursive closure of $\rightarrow_F$ is denoted by $\rightarrow_F^*$.

Now we can define boolean Gröbner bases and characterize them.

Definition 5 *Let I be an ideal of a boolean polynomial ring $B(\bar{X})$. For a subset G of I, we say G is* a boolean Gröbner basis *of I if $\langle lm(I) \rangle = \langle lm(G) \rangle$ in $B(\bar{X})$.*

Proposition 6 *Let I be an ideal of boolean polynomial ring $B(\bar{X})$. Then, a finite subset G of I is a boolean Gröbner basis if and only if $h \rightarrow_G^* 0$ for any $h \in I$.*

We remark that, for a given ideal I over an boolean polynomial ring $B(\bar{X})$, reduced boolean Gröbner basis G of I is not uniquely determined in general.

Definition 7 *A reduced boolean Gröbner basis G is said to be* stratified *if G does not contain two boolean polynomials which have the same leading term.*

Proposition 8 *If G and G' are stratified boolean Gröbner bases of the same ideal with respect to the same term order, then $G = G'$.*

3 Matrices over boolean algebra

Fix a term order $\prec$ on $T(\bar{X})$. Let $t_1, \ldots, t_N$ be the enumeration of $BT(\bar{X})$ such that $t_1 \prec \cdots \prec t_N$, and so $N = 2^n$. Then there is the canonical isomorphism $\psi \colon B(\bar{X}) \rightarrow B^N$ such that $\psi(b_1 t_1 + \cdots + b_N t_N) = (b_1, \ldots, b_N)$ where $b_1, \ldots, b_N \in B$. Identifying boolean polynomials with vectors over boolean ring in this way, we give a procedure to compute boolean Gröbner bases within a linear algebra over a boolean ring. The algorithm is a minor modification of the one of [13], and their essential part is in RREF (Reduced Row Echelon Form) and the function mult().

First we give a definition of RREF of a matrix over a boolean ring, which is a minor modification of the one over a field.

Definition 9 *Let M be a matrix over a boolean ring B. We say M is in* Reduced Row Echelon Form (RREF) *if the followings are satisfied:*

1. *all nonzero rows are above any rows of all zeros,*

2. *the leading coefficient (pivot) of a nonzero row is always to the right of the leading coefficient of the row above it,*

3. *every leading coefficient is the only nonzero entry in its column, and*

4. *every row r is boolean-closed, i.e. $hc(r) \cdot r = r$.*

From a given matrix over boolean ring, several algorithms to compute RREF exists. For example, we can use modified Gaussian Elimination to compute them. We let M be the processing matrix. During the Gaussian Elimination procedure, for each row r, if r is not boolean closed, we replace the row r with $hc(r) \cdot r$ and append the row $r + hc(r) \cdot r$ to M if it is nonzero. Moreover, if there are two rows r_1 and r_2 of M whose leading coefficients b_1 and b_2 respectively are on the same column with $b_1 \cdot b_2 = 0$ and both of b_1 and b_2 are boolean-closed we replace the row r_1 with $r_1 + r_2$ and delete the row r_2. Then, after the modified Gaussian Elimination procedure, we get the RREF of the given matrix. We also notice that this RREF procedure can be computed within at most N-many rows, so we does not need additional memory allocation if we reserve N-many rows at the beginning of the computation.

For the function mult(), we should apply minor modification to the one of [13]. For two boolean power product $\alpha = X_1^{l_1} \cdots X_n^{l_n}$ and $\beta = X_1^{m_1} \cdots X_n^{m_n}$, we define $\alpha * \beta$ by

$$\alpha * \beta = X_1^{\min\{1, l_1+m_1\}} \cdots X_n^{\min\{1, l_n+m_n\}}.$$

Then we see that $\alpha * \beta$ becomes a boolean power product again. For boolean polynomial $f \in B(\bar{X})$ and a boolean power product $\alpha \in BT(\bar{X})$, saying that $f = b_1 t_1 + \cdots + b_N t_N$ where $b_1, \ldots, b_N \in B$, we define $f * \alpha = b_1(t_1 * \alpha) + \cdots + b_N(t_N * \alpha) \in B(\bar{X})$ canonically.

Notation 10 *For $f \in B(\bar{X})$, we define* $\mathrm{mult}(f) \subseteq B(\bar{X})$ *by*

$$\mathrm{mult}(f) = \{f * \alpha \ : \ \alpha \in BT(\bar{X})\}.$$

For $F \subseteq B(\bar{X})$, we let $\mathrm{mult}(F) = \bigcup_{f \in F} \mathrm{mult}(f)$.

Again, we notice that $\mathrm{mult}(F)$ computation can be done within N-many rows.

4 Main Algorithm and Remarks

Using the notations and definitions above, we can describe the algorithm to compute the stratified boolean Gröbner basis of the ideal generated by the given finite set of boolean polynomials.

Algorithm `BooleanIdealLA`

Input: F : a finite subset of $B(\bar{X})$, $\prec$: a term order on $T(\bar{X})$
Output: G : the stratified boolean Gröbner basis of $\langle F \rangle_{B(\bar{X})}$ with respect to $\prec$

```
F' := ∅;
while F ≠ F' do
    F' := F;
    F := the Reduced Row Echelon Form of F;
    F := mult(F);
```

```
end while
let G be the ⊆-minimal with G ⊆ F and ⟨lm(G)⟩ = ⟨lm(F)⟩;
return G;
end.
```

In this algorithm, we require neither extension of a set of terms nor multi-precision integers, besides the one of [13] requires them. So, during the computation, additional memory allocation does not occur, though a mass of memory is required at the beginning when the number of variables is large. Thus, high-speed computation of stratified boolean Gröbner bases can be expected using this algorithm in some cases.

References and Notes

[1] Faugère, J.C. *A new efficient algorithm for computing Gröbner bases (F_4).* J. Pure and Applied Algebra, **139**(1–3), 61–88, (1999)

[2] Faugère, J.C. *A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5).* Proc. ISSAC '02, 75–83, (2002)

[3] Inoue, S. *On the Computation of Comprehensive Boolean Gröbner Bases.* 11th CASC Proceedings, LNCS 5743 (2009)

[4] Kandri-Rody, A, Kapur, D. and Narendran, P. *An Ideal-Theoretic Approach for Word Problems and Unification Problems over Commutative Algebras* Proc RTA-85, LNCS 202, 345-364, (1985)

[5] Kapur, D. and Narendran, P. *An Equational Approach to Theorem Proving in First-Order Predicate Calculus* Proc. IJCAI-85, 1146–1153, (1985)

[6] Lazard, D. *Gröbner-Bases, Gaussian elimination and resolution of systems of algebraic equations.* EUROCAL 1983, 146-156, (1983)

[7] Sakai, K. and Sato, Y. *Boolean Gröbner bases.* ICOT Technical Memorandum 488. (1988)
`http://www.icot.or.jp/ARCHIVE/Museum/TRTM/tm-list-E.html`

[8] Sakai, K., Sato, Y. and Menju, S. *Boolean Gröbner bases(revised).* ICOT Technical Report 613. (1991)
`http://www.icot.or.jp/ARCHIVE/Museum/TRTM/tr-list-E.html`

[9] Sato, Y. *A new type of canonical Gröbner bases in polynomial rings over Von Neumann regular rings.* Proceedings of ISSAC 1998, ACM Press, 317–32, (1998)

[10] Sato, Y., Inoue, S., Suzuki, A. and Nabeshima, K. *Boolean Gröbner Bases and Sudoku.* JSC special issue on Gröbner Bases and Applications, submitting

[11] Sato, Y., Nagai, A. and Inoue, S. *On the Computation of Elimination Ideals of Boolean Polynomial Rings.* LNCS 5081, 338–348.

[12] Suzuki, A. and Sato, Y. (2003). An Alternative approach to Comprehensive Gröbner Bases. J. Symb. Comp. 36/3-4, 649–667.

[13] Suzuki, A. *Computing Gröbner Bases within Linear Algebra.* 11th CASC Proceedings, LNCS 5743, 310–321, (2009)

[14] Weispfenning, V. *Gröbner bases in polynomial ideals over commutative regular rings.* In Davenport Ed., editor, EUROCALf87, Springer LNCS 378, 336–347, (1989)

Finite Element Time Domain Method for Electromagnetic Wave Problems

Kengo Taira[1*] and Seiji Fujino[2]

[1] Graduate School of Information Science and Electrical Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
E-mail: qqkw3u2d@cube.ocn.ne.jp

[2] Research Institute for Information Technology, Kyushu University,
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
E-mail: fujino@cc.kyushu-u.ac.jp

Abstract

The finite element time domain method for electromagnetic wave problems is investigated. We use Galerkin's method to discretize Maxwell's scalar equations directly in the spatial domain. Assembling finite element equations, the system of equations becomes a first-order differential equation with respect to time. Crank-Nicolson's method and Gear's method are used to solve the equation, and to solve the linear system of equations, IDR(s) iterative method is used. As numerical examples, we compute wave propagations in a two dimensional rectangular wave guide with an inserted dielectric obstacle, and wave excitation in a circular resonator.

1 Introduction

In the field of numerical analysis of electromagnetic field, FEM is one of principal methods. FEM is a powerful and versatile numerical technique for handling wave guides with complicated geometric shapes, and inhomogeneous or anisotropic media. FEM is expected to be useful for the analysis of electromagnetic field problems in the time domain but there is few report on the finite element method in the time domain.

The purpose of this work is to develop Finite Element Time Domain Method (FETD) for electromagnetic wave problems. In the finite element method of electromagneticic wave problems, the conventional approach is to impose stationary requirements on the functional for the wave equation derived from Maxwell's equations. It is difficult to introduce the functional of the wave equations in the time domain problems under given boundary conditions.

The formulation using Galerkin's method is available. After discretizing all scalar equations of Maxwell's equations directly in the spatial domain by Galerkin's method, the system of equations assembled from finite element equations becomes a first-order differential equation with respect to time. Crank-Nicolson's method and Gear's method are used to solve the equation. For solving the linear system of equations, IDR(s) method based on IDR (Induced Dimension Reduction) Theorem is used. The formulation of FETD is so simple that this method can be widely adopted in electromagnetic field analyses as Finite Difference Time Domain Method (FDTD).

*(Home) 5-24-21, Nishihara, Urasoe-shi, Okinawa 901-2101, Japan, Tel: 080-27156273.

2 Formulation of FETD

2.1 Deriviation of Finite Element Equations from Maxwell's Eqn.

We consider Maxwell's equations

$$\begin{cases} \nabla \times \mathbf{H} = \varepsilon \frac{\partial \mathbf{E}}{\partial t} + \mathbf{J}, \\ \nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} \end{cases} \tag{1}$$

combined with the boundary conditions and initial conditions in the solution region Ω. In equations (1), $\mathbf{E}$ is the electric field intensity, $\mathbf{H}$ is the magnetic field intensity, $\mathbf{J}$ is an electric current density wave source and ε, μ are permittivity and permeability of the isotropic medium respectively. The region Ω is discretized into finite number of subregions, denoted as Ω_e, and the electromagnetic fields at time t in the element region are approximated by interpolation functions and the values of fields at nodal points. The x component of the electric field at any point in the element Ω_e is expressed in the following form: $E_x(x,y,z,t) = \{N(x,y,z)\}^T\{E_x(t)\}$, $\{N(x,y,z)\}^T = \{N_1, N_2, \cdots, N_m\}$ is the raw vector of the interpolation functions, and $\{E_x(t)\}^T = \{E_1, E_2, \cdots, E_m\}$ is the raw vector of the values of electric field component at nodes, where m is the number of nodes in the element Ω_e. Here, T denotes the transpose of a vector. Galerkin's method is applied for derivation of finite element equations from Maxwell's equations. For example, the column vector $\{N\}$ is multiplied to the residual of the x component of Ampere's law equation of Eqs. (1) and integrated over the element Ω_e, the following equation is obtained:

$$\int_{\Omega_e} \{N\}\left(\frac{\partial E_x}{\partial t} - \frac{1}{\varepsilon}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z}\right) - \frac{1}{\varepsilon} J_x\right) dv = \{0\}. \tag{2}$$

Substituting $E_x = \{N\}^T\{E_x\}$ and similar expressions of H_z, H_y into Eq.(2), yields the finite element equation:

$$[M]_e \frac{d\{E_x\}}{dt} = \frac{1}{\varepsilon}[B]_e\{H_z\} - \frac{1}{\varepsilon}[C]_e\{H_y\} + \{J_x\}_e. \tag{3}$$

Other finite element equations are derived similarly from components of equations (1). In Eq. (3), the square matrices are given as follows:

$$[M]_e = \iint_{\Omega_e} \{N\}\{N\}^T dxdy, \quad [A]_e = \iint_{\Omega_e} \{N\}\frac{\partial \{N\}^T}{\partial x} dxdy,$$

$$[B]_e = \iint_{\Omega_e} \{N\}\frac{\partial \{N\}^T}{\partial y} dxdy, \quad [C]_e = \iint_{\Omega_e} \{N\}\frac{\partial \{N\}^T}{\partial z} dxdy,$$

and

$$\{\mathbf{J}\}_e = \int_{\Omega_e} \{N\}\mathbf{J} dv$$

is a current wave source vector in the element.

2.2 The system of equations

The linear system of equations is obtained by assembling the finite element equations which are derived from each component of Maxwell's Equations for all finite elements. The first-order ordinary differential equation in the time domain is given as follows:

$$[P]\frac{d\{u\}}{dt} + [S]\{u\} = \{g(t)\}, \tag{4}$$

where the unknown vector $\{u\}$ is formed so that the values of electromagnetic field components at all nodes in the solution regon are ordered definitely. The vector $\{g(t)\}$ is a wave source assembled current sources. Here, [P] denotes a square band matrix built by $[M]_e$, and [S] is a square matrix constructed from $[A]_e$, $[B]_e$ and $[C]_e$ for all finite elements. In order to solve Eq.(4), Crank-Nicolson's method is used to get the starting value at first step of Gear's Method and then Gear's method, which is known as a stable solution, is used. Taking time step h, Crank-Nicolson's method yields

$$\begin{aligned}([P] + 0.5h[S])\{u\}^{(n)} = ([P] - 0.5h)[S])\{u\}^{(n-1)} \\ +0.5h(\{g\}^{(n-1)} + \{g\}^{(n)}),\end{aligned} \tag{5}$$

and Gear's method gives

$$([P] + \frac{2}{3}h[S])\{u\}^{(n)} = [P](\frac{4}{3}\{u\}^{(n-1)} - \frac{1}{3}\{u\}^{(n-2)}) + \frac{2}{3}h\{g\}^{(n)} \tag{6}$$

In Eqs. (5) and (6), considering the boundary conditions and the given initial condition, the spatial distribution of electromagnetic fields $\{u\}^{(n)}$ at time $t = t_0 + nh$ is calculated successively. For solving the linear system of equations, IDR(s) iterative method[3] is used, in which non-zero entries of the matrices are stored in CCS(Compressed Column Storage) format.

2.3 Boundary Conditions

The boundary conditions at discontinuities are expressed as follows. Tangential **E** on metal walls is zero. Tangential **E** and the normal component of elecric flux density $\mathbf{D} = \varepsilon\mathbf{E}$ are continuous at the interface separating two different media. The boundary condition about magnetic field is satisfied naturally, because the vacuum permeabillity is used. In order to apply boundary conditions, column and raw elements of the coefficient matrix and the raw elements of the right vector in Eqs. (5) and (6) are treated to satisfy the boundary conditions at the common interface of two elements.

2.4 Nonreflecting boundary conditions

When solving open-region problems one needs to truncate the solution region to limit the size of computational domain. This truncation produces artificial boundaries and the reflected waves are generated at the interface. In order to treat nonreflecting boundary condition, attenuated one-directional travelling waves in an absorption region are assumed and finite element equations derived from the wave forms are incorporated into the System of equations. We assume that E_z is described in a following form.

$$E_z(x, y, t) = g(x)e^{-\alpha y}f(y - ct). \tag{7}$$

Where c is the velocity of light, α is an attenuation constant, and $f(y)$, $g(x)$ are arbitrary functions. This formula satisfies the next partial differential equation.

$$\frac{\partial E_z}{\partial t} + c\frac{\partial E_z}{\partial y} + c\alpha E_z = 0. \tag{8}$$

The FETD method is also applied to Eq.(8), and the finite element equation becomes

$$[M]_e \frac{\partial \{E_z\}}{\partial \tau} + ([B]_e + \alpha[M]_e)\{E_z\} = \{0\}, \tag{9}$$

where $\tau = ct$. Element matices are given by previous expressions. Other finite element equations regading to H_x, H_y are derived similarly.

3 Numerical computation and results

3.1 The finite elemet equation for two dimensional waves

In the computation of two dimensional wave guide, we assume that the electric field component E_z and the magnetic field components H_x and H_y do not depend on the z coordinate of a point. The finite elemet equation is shown as follows :

$$\begin{bmatrix} [M]_e & [0] & [M]_e \\ [0] & [M]_e & [0] \\ [0] & [0] & [M]_e \end{bmatrix} \frac{d}{d\tau} \begin{Bmatrix} \{E_z\} \\ \{Z_0 H_x\} \\ \{Z_0 H_y\} \end{Bmatrix} + \begin{bmatrix} [0] & \frac{1}{\varepsilon_r}[B]_e & -\frac{1}{\varepsilon_r}[A]_e \\ [B]_e & [0] & [0] \\ -[A]_e & [0] & [0] \end{bmatrix} \begin{Bmatrix} \{E_z\} \\ \{Z_0 H_x\} \\ \{Z_0 H_y\} \end{Bmatrix} = \begin{Bmatrix} -\frac{Z_0}{\varepsilon_r}\{j_z\}_e \\ \{0\} \\ \{0\} \end{Bmatrix}. \tag{10}$$

In Eq. (10), $Z_0 = \sqrt{\mu_0/\varepsilon_0}$ is the characteristic impeadance in a free space, so the product of Z_0 and H_x and H_y has the same dimension as that of E_z, and ε_r is a relative permittivity. The variable $\tau = ct$ (c: the velocity of light) is used, as an electromagnetic wave travels the distance τ in a time from 0 to t.

3.2 Wave guide models and computational results

3.2.1 Rectangular wave guide with a dielectric post

Fig.1 is a propagation wave guide model in which a TE_{10} wave is incident into a waveguide with a rectangular dielectric post parallel to z direction. The incident TE_{10} wave as an equivalent source is given as follows:

$$E_z = E_0 cos(2\pi f t + \theta)\sin(\frac{\pi}{a}x), \tag{11}$$

where f is frequency and a is a waveguide width. The metal wave guide is terminated by metal walls at both sides. The boundary conditions on metal walls are expressed as the tangential component of $\mathbf{E}$ is zero. The Analytical region of 22.9[mm]×45.8[mm] is divided into rectangular subregions by $N_x = 40$, $N_y = 80$, and 6 nodes triangular finite elements are produced. Frequency = $10GH_z$, the relative permittivity $\varepsilon_r = 6.0$, and a time step $\Delta t = 3.34 \times 10^{-13}$[sec.] are used.

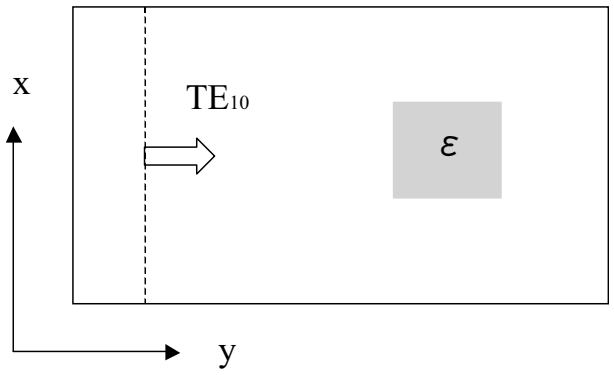


Figure 1: Incident TE_{10} wave in a waveguide with a dielectric rectangular post.

The results of numerical computation are shown in Fig.2. In the figure, time steps are $N = 20, 100, 200, 300, 450$ respectivly. It is shown that TE wave travelling in the wave guide is effected by the dielectric post.

3.2.2 Wave excitation in a circular cavity

Fig.3 (a) shows a circular cavity resonator model and (b) is an its finite descretization. The circular region is divided into 6 nodes triangular finite elements or isoparametric finite elements. The solution region is discretized into 12 divisions in the angular direction of the central circle and 6 divisions in the radial direction. The number of elements is 432 and the number of nodes is 937. On the center of the circle, unit step function current source is impressd. The computational result at time steps $N = 100, 500, 1000, 3000$ is shown in Fig.4. From Fig.4, it is shown that wave is excited and propagates in a cavity resonator.

4 Summary

This paper investigates the finite element time domain (FETD) method in electromagnetic field problems. Using Galerkin's method Maxwell's scalar equations are discretized directly in the spatial domain and the first-order ordinary differential equation in the time domain is solved. To solve the system of equations Crank-Nicolson's method and Gear's method are used. IDR(s) iterative method is used to solve the resulting linear system of equations.

As numerical examples, we study electromagnetic field propagations in a two dimensional waveguide with a dielectric post and a wave excitation in a circular cavity resonator are investigated. The obtained results show that FETD method proved to be useful for the wave guide problems and is capable of providing a versatile approach to electromagnetic wave problems. In order to treat nonreflecting boundary condition, we assume the attenuated one-directional travelling waves in an absorption region and incorporate the finite element equations into the system of equations.

We are going to apply this approach to two dimential waveguides and we will report results in the ASCM conference, as computations are not carried out at this moment.

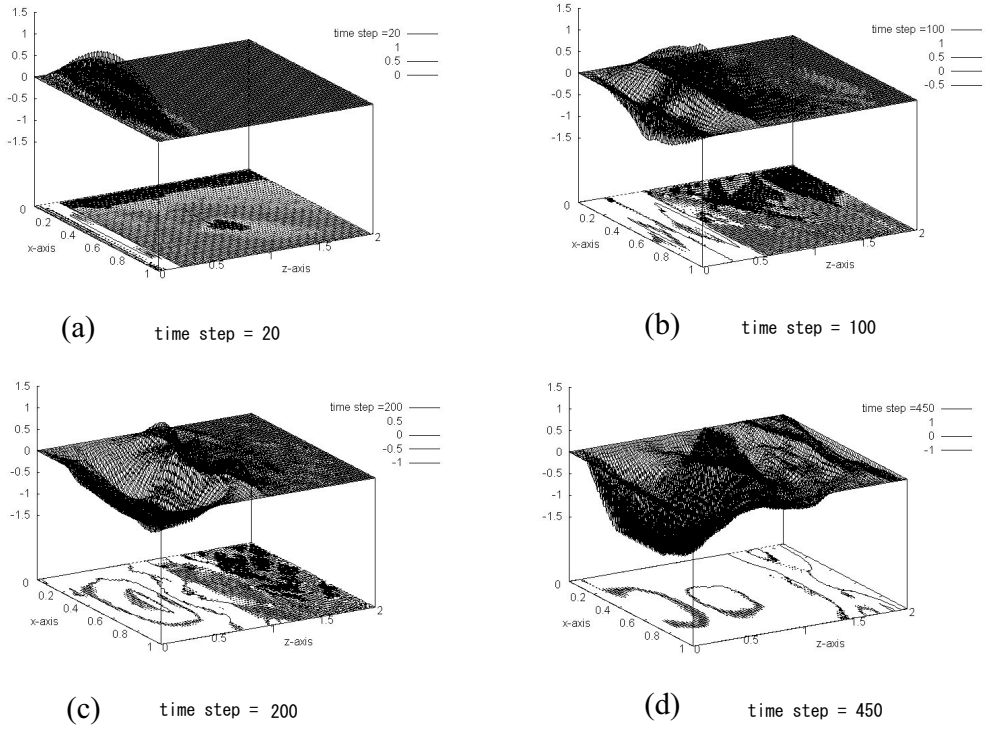


Figure 2: Electric field distribution in a wave guide with a dielectric post.

References and Notes

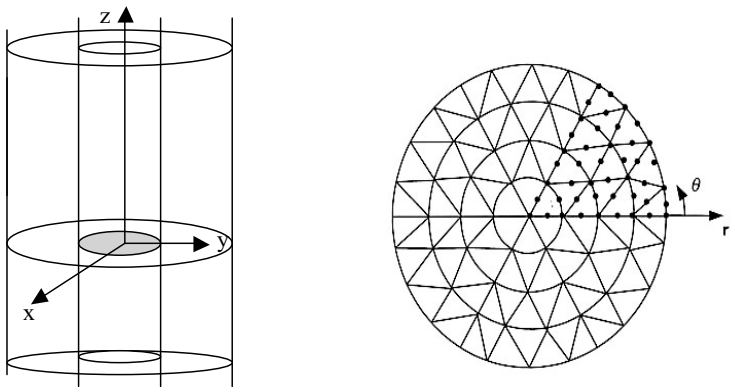
[1] K. Taira, "An approach for solving Maxwell's equations by Finite Element Time Domain Method," The Proc. of the 2001 Electronics Society Conference of IEICE, p.6, 2001.

[2] K. Taira, S. Fujino, "Solution of Maxwell's Equations on Wave Guides by FETD," 2007 Japan-Indo Workshop on Microwaves, Photonics, and Communication Systems at Kyushu University, pp.146-151, July, 2007.

[3] Y. Onoue, S. Fujino, N. Nakashima, "Outline of IDR(s) algorithm and convergence evaluation ," The 2007 annual meeting of the Japan Society for Industrial and Applied Mathematics, pp.126-127, Sept., 2007.

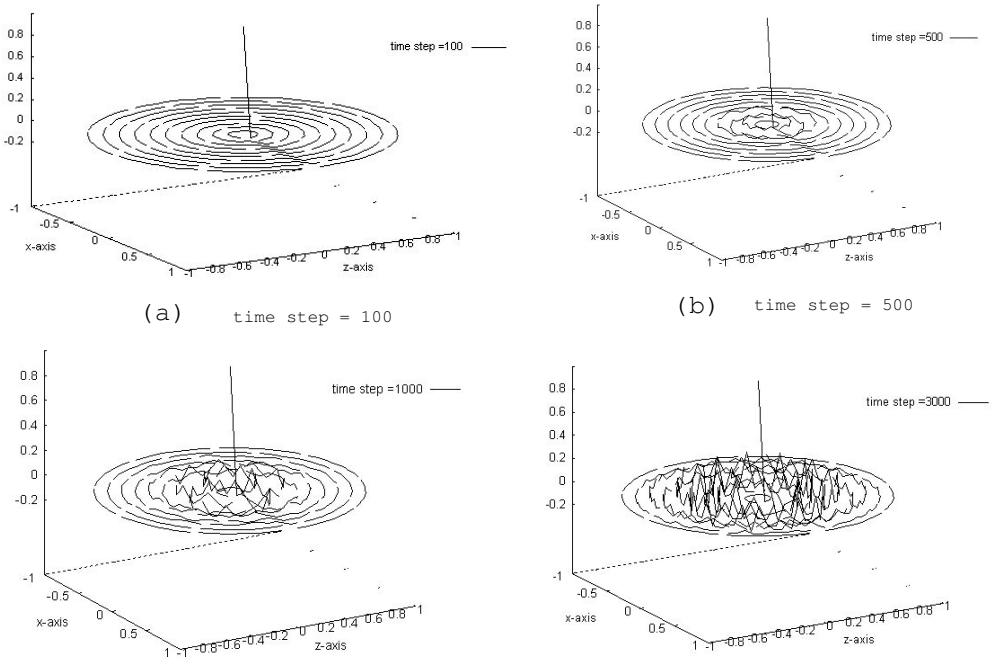
[4] K. Taira, S. Fujino, "An FETD Method for Wave guide Problems," The Proc. of 2008 China-Japan joint Microwave Conference, Vol.1, pp.40-43, Sept., 2008.

[5] K. Taira, S. Fujino, "Electromagnetic Wave Propagation Simulation in Wave-Guides by Two Dimensional FETD method," The Proc. of the 28th Japan Simulation Conference, pp.357-360, June, 2009.



(a) (b)

Figure 3: (a) Circular resonator model; (b) its finite element descretization.



(a) time step = 100 (b) time step = 500

(c) time step = 1000 (d) time step = 3000

Figure 4: Electric field distribution in a circular resonator.

GPGCD, an Iterative Method for Calculating Approximate GCD of Univariate Polynomials, with the Complex Coefficients

Akira Terui

Graduate School of Pure and Applied Sciences
University of Tsukuba
Tsukuba, 305-8571 Japan
terui@math.tsukuba.ac.jp

Abstract

We present an extension of our GPGCD method, an iterative method for calculating approximate greatest common divisor (GCD) of univariate polynomials, to polynomials with the complex coefficients. For a given pair of polynomials and a degree, our algorithm finds a pair of polynomials which has a GCD of the given degree and whose coefficients are perturbed from those in the original inputs, making the perturbations as small as possible, along with the GCD. In our GPGCD method, the problem of approximate GCD is transfered to a constrained minimization problem, then solved with a so-called modified Newton method, which is a generalization of the gradient-projection method, by searching the solution iteratively. While our original method is designed for polynomials with the real coefficients, we extend it to accept polynomials with the complex coefficients in this paper.

1 Introduction

For algebraic computations on polynomials and matrices, approximate algebraic algorithms are attracting broad range of attentions recently. These algorithms take inputs with some "noise" such as polynomials with floating-point number coefficients with rounding errors, or more practical errors such as measurement errors, then, with minimal changes on the inputs, seek a meaningful answer that reflect desired property of the input, such as a common factor of a given degree. By this characteristic, approximate algebraic algorithms are expected to be applicable to more wide range of problems, especially those to which exact algebraic algorithms were not applicable.

As an approximate algebraic algorithm, we consider calculating the approximate greatest common divisor (GCD) of univariate polynomials, such that, for a given pair of polynomials and a degree d, finding a pair of polynomials which has a GCD of degree d and whose coefficients are perturbations from those in the original inputs, with making the perturbations as small as possible, along with the GCD. This problem has been extensively studied with various approaches including the Euclidean method on the polynomial remainder sequence (PRS) ([1], [14], [15]), the singular value decomposition (SVD) of the Sylvester matrix ([3], [6]), the QR factorization of the Sylvester matrix or its displacements ([4], [18], [20]), Padé approximation [11], optimization strategies ([2], [7], [8], [9], [19]). Furthermore, stable methods for ill-conditioned problems have been discussed ([4], [10], [13]).

Among methods in the above, we focus our attention on optimization strategies. Already proposed algorithms utilize iterative methods including the Levenberg-Marquardt method [2], the Gauss-Newton method [19] and the structured total least norm (STLN) method ([7], [8]). Among them, STLN-based methods have shown good performance calculating approximate GCD with sufficiently small perturbations efficiently.

In this paper, we discuss an extension of the GPGCD method, proposed by the present author [17], an iterative method with transferring the original approximate GCD problem into a constrained optimization problem, then solving it by a so-called modified Newton method [16], which is a generalization of the gradient-projection method [12]. In the previous paper [17], we have shown that our method calculates approximate GCD with perturbations as small as those calculated by the STLN-based methods and with significantly better efficiency than theirs. While our original method accepts polynomials with the real coefficients as inputs and outputs in the previous paper, we extend it to handle polynomials with the complex coefficients in more generalized settings in this paper.

The rest part of the paper is organized as follows. In Section 2, we transform the approximate GCD problem into a constrained minimization problem for the case with the complex coefficients. In Section 3, we show details for calculating the approximate GCD, with discussing issues in minimizations. In Section 4, we demonstrate performance of our algorithm with experiments.

2 Formulation of the Approximate GCD Problem

Let $F(x)$ and $G(x)$ be univariate polynomials of degree m and n, respectively, with the complex coefficients and $m \geq n > 0$. We permit F and G be relatively prime in general. For a given integer d satisfying $n \geq d > 0$, let us calculate a deformation of $F(x)$ and $G(x)$ in the form of

$$\tilde{F}(x) = F(x) + \Delta F(x) = H(x) \cdot \bar{F}(x), \quad \tilde{G}(x) = G(x) + \Delta G(x) = H(x) \cdot \bar{G}(x),$$

where $\Delta F(x)$ and $\Delta G(x)$ are polynomials with the complex coefficients, whose degrees do not exceed those of $F(x)$ and $G(x)$, respectively, $H(x)$ is a polynomial of degree d, and $\bar{F}(x)$ and $\bar{G}(x)$ are pairwise relatively prime. In this situation, $H(x)$ is an approximate GCD of $F(x)$ and $G(x)$. For a given d, we try to minimize $\|\Delta F(x)\|_2^2 + \|\Delta G(x)\|_2^2$ the norm of the deformations.

In the case $\tilde{F}(x)$ and $\tilde{G}(x)$ have a GCD of degree d, then the theory of subresultant tells us that the $(d-1)$-th subresultant of $\tilde{F}$ and $\tilde{G}$ becomes zero, namely we have $\mathrm{S}_{d-1}(\tilde{F}, \tilde{G}) = 0$, where $\mathrm{S}_k(\tilde{F}, \tilde{G})$ denotes the k-th subresultant of $\tilde{F}$ and $\tilde{G}$. Then, the $(d-1)$-th subresultant matrix $N_{d-1}(F, G)$, where the k-th subresultant matrix $N_k(F, G)$ is a submatrix of the Sylvester matrix $N(F, G)$ by taking the left $n-k$ columns of coefficients of F and the left $m-k$ columns of coefficients of G, has a kernel of dimension equal to 1. Thus, there exist polynomials $A(x), B(x) \in \mathbf{C}[x]$ satisfying

$$A\tilde{F} + B\tilde{G} = 0, \tag{1}$$

with $\deg(A) < n-d$ and $\deg(B) < m-d$ and $A(x)$ and $B(x)$ are relatively prime. Therefore, for the given $F(x)$, $G(x)$ and d, our problem is to find $\Delta F(x)$, $\Delta G(x)$, $A(x)$ and $B(x)$ satisfying Eq. (1) with making $\|\Delta F\|_2^2 + \|\Delta G\|_2^2$ as small as possible.

Let us assume that $F(x)$ and $G(x)$ are represented as

$$\begin{aligned} F(x) &= (f_{m,1} + f_{m,2}\boldsymbol{i})x^m + \cdots + (f_{0,1} + f_{0,2}\boldsymbol{i}) = F_{\mathrm{Re}}(x) + \boldsymbol{i}F_{\mathrm{Im}}(x), \\ G(x) &= (g_{n,1} + g_{n,2}\boldsymbol{i})x^n + \cdots + (g_{0,1} + g_{0,2}\boldsymbol{i}) = G_{\mathrm{Re}}(x) + \boldsymbol{i}G_{\mathrm{Im}}(x), \end{aligned}$$

where $f_{j,1}$, $g_{j,1}$, $f_{j,2}$, $g_{j,2}$ are the real numbers and $\boldsymbol{i}$ is the imaginary unit, and $F_{\mathrm{Re}}(x)$ and $G_{\mathrm{Re}}(x)$ represent the real part of $F(x)$ and $G(x)$, respectively, while $F_{\mathrm{Im}}(x)$ and $G_{\mathrm{Im}}(x)$

represent the imaginary part of $F(x)$ and $G(x)$, respectively. Furthermore, we represent $\tilde{F}(x)$, $\tilde{G}(x)$, $A(x)$ and $B(x)$ with the complex coefficients as

$$\begin{aligned}
\tilde{F}(x) &= (\tilde{f}_{m,1} + \tilde{f}_{m,2}\boldsymbol{i})x^m + \cdots + (\tilde{f}_{0,1} + \tilde{f}_{0,2}\boldsymbol{i})x^0 = \tilde{F}_{\mathrm{Re}}(x) + \boldsymbol{i}\tilde{F}_{\mathrm{Im}}(x),\\
\tilde{G}(x) &= (\tilde{g}_{n,1} + \tilde{g}_{n,2}\boldsymbol{i})x^n + \cdots + (\tilde{g}_0 x^0 + \tilde{g}_{0,2}\boldsymbol{i})x^0 = \tilde{G}_{\mathrm{Re}}(x) + \boldsymbol{i}\tilde{G}_{\mathrm{Im}}(x),\\
A(x) &= (a_{n-d,1} + a_{n-d,2}\boldsymbol{i})x^{n-d} + \cdots + (a_{0,1} + a_{0,2}\boldsymbol{i})x^0 = A_{\mathrm{Re}}(x) + \boldsymbol{i}A_{\mathrm{Im}}(x),\\
B(x) &= (b_{m-d,1} + b_{m-d,2}\boldsymbol{i})x^{m-d} + \cdots + (b_{0,1} + b_{0,2}\boldsymbol{i})x^0 = B_{\mathrm{Re}}(x) + \boldsymbol{i}B_{\mathrm{Im}}(x),
\end{aligned} \tag{2}$$

respectively, where $\tilde{f}_{j,1}$, $\tilde{f}_{j,2}$, $\tilde{g}_{j,1}$, $\tilde{g}_{j,2}$, $a_{j,1}$, $a_{j,2}$, $b_{j,1}$, $b_{j,2}$ are the real numbers, and, as in above, $\tilde{F}_{\mathrm{Re}}(x)$, $\tilde{G}_{\mathrm{Re}}(x)$, $A_{\mathrm{Re}}(x)$ and $B_{\mathrm{Re}}(x)$ represent the real part of $\tilde{F}(x)$, $\tilde{G}(x)$, $A(x)$ and $B(x)$, respectively, while $\tilde{F}_{\mathrm{Im}}(x)$, $\tilde{G}_{\mathrm{Im}}(x)$, $A_{\mathrm{Im}}(x)$ and $B_{\mathrm{Im}}(x)$ represent the imaginary part of $\tilde{F}(x)$, $\tilde{G}(x)$, $A(x)$ and $B(x)$, respectively.

For the objective function, $\|\Delta F\|_2^2 + \|\Delta G\|_2^2$ becomes as

$$\sum_{j=0}^{m}[(\tilde{f}_{j,1} - f_{j,1})^2 + (\tilde{f}_{j,2} - f_{j,2})^2] + \sum_{j=0}^{n}[(\tilde{g}_{j,1} - g_{j,1})^2 + (\tilde{g}_{j,2} - g_{j,2})^2]. \tag{3}$$

For the constraint, Eq. (1) becomes as

$$N_{d-1}(\tilde{F}, \tilde{G}) \cdot {}^t\left(a_{n-d,1} + a_{n-d,2}\boldsymbol{i}, \ldots, a_{0,1} + a_{0,2}\boldsymbol{i}, b_{m-d,1} + b_{m-d,2}\boldsymbol{i}, \ldots, b_{0,1} + b_{0,2}\boldsymbol{i}\right) = \boldsymbol{0}. \tag{4}$$

By expressing the subresultant matrix and the column vector in (4) separated into the real and the complex parts, respectively, we express (4) as

$$(N_1 + N_2\boldsymbol{i})(\boldsymbol{v}_1 + \boldsymbol{v}_2\boldsymbol{i}) = \boldsymbol{0}, \tag{5}$$

$$\begin{aligned}
&N_1 = N_{d-1}(\tilde{F}_{\mathrm{Re}}(x), \tilde{G}_{\mathrm{Re}}(x)), \quad N_2 = N_{d-1}(\tilde{F}_{\mathrm{Im}}(x), \tilde{G}_{\mathrm{Im}}(x)),\\
&\boldsymbol{v}_1 = {}^t(a_{n-d,1}, \ldots, a_{0,1}, b_{m-d,1}, \ldots, b_{0,1}), \quad \boldsymbol{v}_2 = {}^t(a_{n-d,2}, \ldots, a_{0,2}, b_{m-d,2}, \ldots, b_{0,2}).
\end{aligned} \tag{6}$$

We can expand the left-hand-side of Eq. (5) as $(N_1 + N_2\boldsymbol{i})(\boldsymbol{v}_1 + \boldsymbol{v}_2\boldsymbol{i}) = (N_1\boldsymbol{v}_1 - N_2\boldsymbol{v}_2) + \boldsymbol{i}(N_1\boldsymbol{v}_2 + N_2\boldsymbol{v}_1)$, thus, Eq. (5) is equivalent to a system of equations: $N_1\boldsymbol{v}_1 - N_2\boldsymbol{v}_2 = \boldsymbol{0}, N_1\boldsymbol{v}_2 + N_2\boldsymbol{v}_1 = \boldsymbol{0}$, which is expressed as

$$\begin{pmatrix} N_1 & -N_2 \\ N_2 & N_1 \end{pmatrix}\begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{pmatrix} = \boldsymbol{0}. \tag{7}$$

Furthermore, we add another constraint for the coefficient of $A(x)$ and $B(x)$ as

$$\begin{aligned}
\|A(x)\|_2^2 + \|B(x)\|_2^2 = {} & (a_{n-d,1}^2 + \cdots + a_{0,1}^2) + (b_{m-d,1}^2 + \cdots + b_{0,1}^2)\\
& + (a_{n-d,2}^2 + \cdots + a_{0,2}^2) + (b_{m-d,2}^2 + \cdots + b_{0,2}^2) - 1 = 0,
\end{aligned} \tag{8}$$

which can be expressed together with (7) as

$$\begin{pmatrix} {}^t\boldsymbol{v}_1 & {}^t\boldsymbol{v}_2 & -1 \\ N_1 & -N_2 & \boldsymbol{0} \\ N_2 & N_1 & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ 1 \end{pmatrix} = \boldsymbol{0}, \tag{9}$$

where Eq. (8) has been put on the top of Eq. (7). Note that, in Eq. (9), we have total of $2(m+n-d+1)+1$ equations in the coefficients of polynomials in (2) as a constraint, with the j-th row of which is expressed as $q_j = 0$.

Now, we substitute the variables

$$(\tilde{f}_{m,1},\ldots,\tilde{f}_{0,1},\tilde{g}_{n,1},\ldots,\tilde{g}_{0,1},\tilde{f}_{m,2},\ldots,\tilde{f}_{0,2},\tilde{g}_{n,2},\ldots,\tilde{g}_{0,2},\\ a_{n-d,1},\ldots,a_{0,1},b_{m-d,1},\ldots,b_{0,1},a_{n-d,2},\ldots,a_{0,2},b_{m-d,2},\ldots,b_{0,2}), \quad (10)$$

as $\boldsymbol{x} = (x_1,\ldots,x_{4(m+n-d+2)})$, then Eq. (3) and (9) become as

$$f(\boldsymbol{x}) = (x_1 - f_{m,1})^2 + \cdots + (x_{m+1} - f_{0,1})^2 + (x_{m+2} - g_{n,1})^2 + \cdots + (x_{m+n+2} - g_{0,1})^2 \\ + (x_{m+n+3} - f_{m,2})^2 + \cdots + (x_{2m+n+3} - f_{0,2})^2 \\ + (x_{2m+n+4} - g_{n,2})^2 + \cdots + (x_{2(m+n+2)} - g_{0,2})^2, \quad (11)$$

$$\boldsymbol{q}(\boldsymbol{x}) = {}^t(q_1(\boldsymbol{x}),\ldots,q_{2(m+n-d+1)+1}(\boldsymbol{x})) = \boldsymbol{0}, \quad (12)$$

respectively. Therefore, the problem of finding an approximate GCD can be formulated as a constrained minimization problem of finding a minimizer of the objective function $f(\boldsymbol{x})$ in Eq. (11), subject to $\boldsymbol{q}(\boldsymbol{x}) = \boldsymbol{0}$ in Eq. (12).

3 The Algorithm for Approximate GCD

We calculate an approximate GCD by solving the constrained minimization problem (11), (12) with the gradient projection method by Rosen [12] (whose initials become the name of our GPGCD method) or a modified Newton method by Tanabe [16] (for review, see the author's previous paper [17]). Our preceding experiments [17, Section 5.1] have shown that a modified Newton method was more efficient than the original gradient projection method while the both methods have shown almost the same convergence property, thus we adopt a modified Newton method in this paper.

In applying a modified Newton method to the approximate GCD problem, we discuss issues in the construction of the algorithm in detail, such as

- Representation of the Jacobian matrix $J_{\boldsymbol{g}}(\boldsymbol{x})$ and certifying that $J_{\boldsymbol{g}}(\boldsymbol{x})$ has full rank (Section 3.1),
- Setting the initial values (Section 3.2),
- Regarding the minimization problem as the minimum distance problem (Section 3.3),
- Calculating the actual GCD and correcting the coefficients of $\tilde{F}$ and $\tilde{G}$ (Section 3.4),

as follows.

3.1 Representation and the rank of the Jacobian Matrix

For a polynomial $P(x) \in \mathbf{C}[x]$ represented as $P(x) = p_n x^n + \cdots + p_0 x^0$, let $C_k(P)$ be a complex $(n+k, k+1)$ matrix defined as

$$C_k(P) = \underbrace{\begin{pmatrix} p_n & & \\ \vdots & \ddots & \\ p_0 & & p_n \\ & \ddots & \vdots \\ & & p_0 \end{pmatrix}}_{k+1}.$$

For co-factors $A(x)$ and $B(x)$ in (2), define matrices A_1 and A_2 as

$$A_1 = [C_m(A_{\mathrm{Re}}(x))\ C_n(B_{\mathrm{Re}}(x))], \quad A_2 = [C_m(A_{\mathrm{Im}}(x))\ C_n(B_{\mathrm{Im}}(x))]. \tag{13}$$

(Note that A_1 and A_2 are matrices of the real numbers of $m+n-d+1$ rows and $m+n+2$ columns.) Then, by the definition of the constraint (12), we have the Jacobian matrix $J_{\boldsymbol{g}}(\boldsymbol{x})$ (with the original notation of variables (10) for $\boldsymbol{x}$) as

$$J_{\boldsymbol{g}}(\boldsymbol{x}) = \begin{pmatrix} \mathbf{0} & \mathbf{0} & 2 \cdot {}^t\boldsymbol{v}_1 & 2 \cdot {}^t\boldsymbol{v}_2 \\ A_1 & -A_2 & N_1 & -N_2 \\ A_2 & A_1 & N_2 & N_1 \end{pmatrix},$$

with A_1 and A_2 as in (13) and N_1, N_2, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ as in (6), respectively, which can be easily constructed in every iteration.

In executing iterations, we need to keep that $J_{\boldsymbol{g}}(\boldsymbol{x})$ has full rank: otherwise, we are unable to decide proper search direction. For this requirement, we have the following observations.

Proposition 1. *Let $\boldsymbol{x}^* \in V_{\boldsymbol{g}}$ be any feasible point satisfying Eq. (12). Then, if the corresponding polynomials do not have a GCD whose degree exceeds d, then $J_{\boldsymbol{g}}(\boldsymbol{x}^*)$ has full rank.*

Proof. Let $\boldsymbol{x}^* = (\tilde{f}_{m,1}, \ldots, \tilde{f}_{0,1}, \tilde{g}_{n,1}, \ldots, \tilde{g}_{0,1}, \tilde{f}_{m,2}, \ldots, \tilde{f}_{0,2}, \tilde{g}_{n,2}, \ldots, \tilde{g}_{0,2}, a_{n-d,1}, \ldots, a_{0,1}, b_{m-d,1}, \ldots, b_{0,1}, a_{n-d,2}, \ldots, a_{0,2}, b_{m-d,2}, \ldots, b_{0,2})$ with its polynomial representation expressed as in (2) (note that this assumption permits the polynomials $\tilde{F}(x)$ and $\tilde{G}(x)$ to be relatively prime in general). To verify our claim, we show that we have $\mathrm{rank}(J_{\boldsymbol{g}}(\boldsymbol{x}^*)) = 2(m+n-d+1)+1$. Let us express $J_{\boldsymbol{g}}(\boldsymbol{x}^*) = \left(J_{\mathrm{L}} \mid J_{\mathrm{R}}\right)$, where J_{L} and J_{R} are column blocks expressed as

$$J_{\mathrm{L}} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ A_1 & -A_2 \\ A_2 & A_1 \end{pmatrix}, \quad J_{\mathrm{R}} = \begin{pmatrix} 2 \cdot {}^t\boldsymbol{v}_1 & 2 \cdot {}^t\boldsymbol{v}_2 \\ N_1 & -N_2 \\ N_2 & N_1 \end{pmatrix},$$

respectively. Then, we have the following lemma.

Lemma 1. *We have* $\mathrm{rank}(J_{\mathrm{L}}) = 2(m+n-d+1)$.

Proof. For $A_1 = [C_m(A)_1\ C_n(B)_1]$, let $\overline{C_m(A)_1}$ be the right $m-d$ columns of $C_m(A)_1$ and $\overline{C_n(B)_1}$ be the right $n-d$ columns of $C_n(B)_1$. Then, we see that the bottom $m+n-2d$ rows of the matrix $\bar{C} = [\overline{C_m(A)_1}\ \overline{C_n(B)_1}]$ is equal to the matrix consisting of the real part of the elements of $N(A,B)$, the Sylvester matrix of $A(x)$ and $B(x)$. By the assumption, polynomials $A(x)$ and $B(x)$ are relatively prime, and there exist no nonzero elements in $\bar{C}$ except for the bottom $m+n-2d$ rows, thus we have $\mathrm{rank}(\bar{C}) = m+n-2d$.

By the structure of $\bar{C}$ and the lower triangular structure of $C_m(A)_1$ and $C_n(B)_1$, we can take the left $d+1$ columns of $C_m(A)_1$ or $C_n(B)_1$ satisfying linear independence along with $\bar{C}$, which implies that there exist a nonsingular square matrix T of order $m+n+2$ satisfying

$$A_1 T = R, \tag{14}$$

where R is a lower triangular matrix, thus we have $\mathrm{rank}(A_1) = \mathrm{rank}(R) = m+n-d+1$.

Furthermore, by using T and R in (14), we have

$$\begin{pmatrix} \mathbf{0} & \mathbf{0} \\ A_1 & -A_2 \\ A_2 & A_1 \end{pmatrix} \begin{pmatrix} T & \mathbf{0} \\ \mathbf{0} & T \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ R & -A_2T \\ A_2T & R \end{pmatrix}, \tag{15}$$

followed by a suitable transformation on columns on the matrix in the right-hand-side of (15), we can make A_2T to zero matrix, which implies that

$$\operatorname{rank}(J_{\mathrm{L}}) = \operatorname{rank}\left(\begin{pmatrix} \mathbf{0} & \mathbf{0} \\ R & -A_2T \\ A_2T & R \end{pmatrix}\right) = 2 \cdot \operatorname{rank}(R) = 2(m+n-d+1).$$

This proves the lemma. □

Proof of Proposition 1 (continued). By the assumptions, we have at least one nonzero coordinate in the top row in J_{R}, while we have no nonzero coordinate in the top row in J_{L}, thus we have $\operatorname{rank}(J_{\boldsymbol{g}}(\boldsymbol{x})) = 2(m+n-d+1)+1$, which proves the proposition. □

Proposition 1 says that, so long as the search direction in the minimization problem satisfies that corresponding polynomials have a GCD of degree not exceeding d, then $J_{\boldsymbol{g}}(\boldsymbol{x})$ has full rank, thus we can safely calculate the next search direction for approximate GCD.

3.2 Setting the Initial Values

At the beginning of iterations, we give the initial value $\boldsymbol{x}_0$ by using the singular value decomposition (SVD) [5] of $N = \begin{pmatrix} N_1 & -N_2 \\ N_2 & N_1 \end{pmatrix}$ in (7) as $N = U\,\Sigma\,{}^tV, U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{2(m+n-2d+2)})$, $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_{2(m+n-2d+2)}), V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{2(m+n-2d+2)})$, with $\boldsymbol{u}_j \in \mathbf{R}^{2(m+n-d+1)}$, $\boldsymbol{v}_j \in \mathbf{R}^{2(m+n-2d+2)}$, and $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_{2(m+n-2d+2)})$ denotes the diagonal matrix whose the j-th diagonal element is σ_j. Note that U and V are orthogonal matrices. Then, by a property of the SVD [5, Theorem 3.3], the smallest singular value $\sigma_{2(m+n-2d+2)}$ gives the minimum distance of the image of the unit sphere $\mathrm{S}^{2(m+n-2d+2)-1}$, given as $\mathrm{S}^{2(m+n-2d+2)-1} = \{\boldsymbol{x} \in \mathbf{R}^{2(m+n-2d+2)} \mid \|\boldsymbol{x}\|_2 = 1\}$, by N, represented as $N \cdot \mathrm{S}^{2(m+n-2d+1)-1} = \{N\boldsymbol{x} \mid \boldsymbol{x} \in \mathbf{R}^{2(m+n-2d+2)}, \|\boldsymbol{x}\|_2 = 1\}$, from the origin, along with $\sigma_{2(m+n-2d+2)}\boldsymbol{u}_{2(m+n-2d+2)}$ as its coordinates. Thus, we have $N \cdot \boldsymbol{v}_{2(m+n-2d+2)} = \sigma_{2(m+n-2d+2)}\boldsymbol{u}_{2(m+n-2d+2)}$. For $\boldsymbol{v}_{m+n-2d} = {}^t(\bar{a}_{n-d}, \ldots, \bar{a}_0, \bar{b}_{n-d}, \ldots, \bar{b}_0)$, let $\bar{A}(x) = \bar{a}_{n-d}x^{n-d} + \cdots + \bar{a}_0x^0$ and $\bar{B}(x) = \bar{b}_{m-d}x^{m-d} + \cdots + \bar{b}_0x^0$. Then, $\bar{A}(x)$ and $\bar{B}(x)$ give the least norm of $AF + BG$ satisfying $\|A\|_2^2 + \|B\|_2^2 = 1$ by putting $A(x) = \bar{A}(x)$ and $B(x) = \bar{B}(x)$ in (2).

Therefore, we admit the coefficients of F, G, $\bar{A}$ and $\bar{B}$ as the initial values of the iterations as

$$\begin{aligned}\boldsymbol{x}_0 = (&f_{m,1}, \ldots, f_{0,1}, g_{n,1}, \ldots, g_{0,1}, f_{m,2}, \ldots, f_{0,2}, g_{n,2}, \ldots, g_{0,2}, \\ &\bar{a}_{n-d,1}, \ldots, \bar{a}_{0,1}, \bar{b}_{n-d,1}, \ldots, \bar{b}_{0,1}, \bar{a}_{n-d,2}, \ldots, \bar{a}_{0,2}, \bar{b}_{n-d,2}, \ldots, \bar{b}_{0,2}).\end{aligned}$$

3.3 Regarding the Minimization Problem as the Minimum Distance (Least Squares) Problem

Since we have the object function f as in (11), we have

$$\nabla f(\boldsymbol{x}) = 2 \cdot {}^t(x_1 - f_{m,1}, \ldots, x_{m+1} - f_{0,1}, x_{m+2} - g_{n,1}, \ldots, x_{m+n+2} - g_{0,1},$$
$$x_{m+n+3} - f_{m,2}, \ldots, x_{2m+n+3} - f_{0,2}, x_{2m+n+4} - g_{n,2}, \ldots, x_{2(m+n+2)} - g_{0,2}, 0, \ldots, 0).$$

However, we can regard our problem as finding a point $\boldsymbol{x} \in V_{\boldsymbol{g}}$ which has the minimum distance to the initial point $\boldsymbol{x}_0$ with respect to the $(x_1, \ldots, x_{2(m+n+2)})$-coordinates which correspond to the coefficients in $F(x)$ and $G(x)$. Therefore, as in the real case (see the authors previous paper [17]), we change the objective function as $\bar{f}(\boldsymbol{x}) = \frac{1}{2} f(\boldsymbol{x})$, then solve the minimization problem of $\bar{f}(\boldsymbol{x})$, subject to $\boldsymbol{q}(\boldsymbol{x}) = \boldsymbol{0}$.

3.4 Calculating the Actual GCD and Correcting the Deformed Polynomials

After successful end of the iterations, we obtain the coefficients of $\tilde{F}(x)$, $\tilde{G}(x)$, $A(x)$ and $B(x)$ satisfying (1) with $A(x)$ and $B(x)$ are relatively prime. Then, we need to compute the actual GCD $H(x)$ of $\tilde{F}(x)$ and $\tilde{G}(x)$. Although H can be calculated as the quotient of $\tilde{F}$ divided by B or $\tilde{G}$ divided by A, naive polynomial division may cause numerical errors in the coefficient. Thus, we calculate the coefficients of H by the so-called least squares division [19], followed by correcting the coefficients in $\tilde{F}$ and $\tilde{G}$ by using the calculated H, as follows.

For polynomials $\tilde{F}$, $\tilde{G}$, A and B represented as in (2) and H represented as $H(x) = (h_{d,1} + h_{d,2}\boldsymbol{i})x^d + \cdots + (h_{0,1} + h_{0,2}\boldsymbol{i})x^0$, solve the equations $HB = \tilde{F}$ and $HA = \tilde{G}$ with respect to H as solving the least squares problems of linear systems

$$C_d(A)\,{}^t(h_{d,1} + h_{d,2}\boldsymbol{i}, \ldots, h_{0,1} + h_{0,2}\boldsymbol{i}) = {}^t(\tilde{g}_{n,1} + \tilde{g}_{n,2}\boldsymbol{i}, \ldots, \tilde{g}_{0,1} + \tilde{g}_{0,2}\boldsymbol{i}), \tag{16}$$
$$C_d(B)\,{}^t(h_{d,1} + h_{d,2}\boldsymbol{i}, \ldots, h_{0,1} + h_{0,2}\boldsymbol{i}) = {}^t(\tilde{f}_{m,1} + \tilde{f}_{m,2}\boldsymbol{i}, \ldots, \tilde{f}_{0,1} + \tilde{f}_{0,2}\boldsymbol{i}), \tag{17}$$

respectively. Then, we transfer the linear systems (16) and (17), as follows. For (17), let us express the matrices and vectors as the sum of the real and the imaginary part of which, respectively, as $C_d(B) = B_1 + \boldsymbol{i}B_2$, ${}^t(h_{d,1} + h_{d,2}\boldsymbol{i}, \ldots, h_{0,1} + h_{0,2}\boldsymbol{i}) = \boldsymbol{h}_1 + \boldsymbol{i}\boldsymbol{h}_2$, ${}^t(\tilde{f}_{m,1} + \tilde{f}_{m,2}\boldsymbol{i}, \ldots, \tilde{f}_{0,1} + \tilde{f}_{0,2}\boldsymbol{i}) = \boldsymbol{f}_1 + \boldsymbol{i}\boldsymbol{f}_2$. Then, (17) is expressed as

$$(B_1 + \boldsymbol{i}B_2)(\boldsymbol{h}_1 + \boldsymbol{i}\boldsymbol{h}_2) = (\boldsymbol{f}_1 + \boldsymbol{i}\boldsymbol{f}_2). \tag{18}$$

By equating the real and the imaginary parts in Eq. (18), respectively, we have $(B_1\boldsymbol{h}_1 - B_2\boldsymbol{h}_2) = \boldsymbol{f}_1, (B_1\boldsymbol{h}_2 + B_2\boldsymbol{h}_1) = \boldsymbol{f}_2$, or

$$\begin{pmatrix} B_1 & -B_2 \\ B_2 & B_1 \end{pmatrix} \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix}. \tag{19}$$

Thus, we can calculate the coefficients of $H(x)$ by solving the real least squares problem (19). We can solve (16) similarly.

Let $H_1(x), H_2(x) \in \mathbf{C}[x]$ be the candidates for the GCD whose coefficients are calculated as the least squares solutions of (16) and (17), respectively. Then, for $i = 1, 2$, calculate the norms of the residues as $r_i = \|\tilde{F} - H_i B\|_2^2 + \|\tilde{G} - H_i A\|_2^2$, respectively, and set the GCD $H(x)$ be $H_i(x)$ giving the minimum value of r_i.

Finally, for the chosen $H(x)$, correct the coefficients of $\tilde{F}(x)$ and $\tilde{G}(x)$ as $\tilde{F}(x) = H(x) \cdot B(x)$, $\tilde{G}(x) = H(x) \cdot A(x)$, respectively.

4 Experiments

We have implemented the GPGCD algorithm for polynomials with the complex coefficients on the computer algebra system Maple and compared its performance with a method based on the structured total least norm (STLN) method [7] for randomly generated polynomials with approximate GCD. The tests have been carried out on Intel Core2 Duo Mobile Processor T7400 (in Apple MacBook "Mid-2007" model) at 2.16 GHz with RAM 2GB, under MacOS X 10.5.

In the tests, we have generated random polynomials with GCD then added noise, as follows. First, we have generated a pair of monic polynomials $F_0(x)$ and $G_0(x)$ of degrees m and n, respectively, with the GCD of degree d. The GCD and the prime parts of degrees $m-d$ and $n-d$ are generated as monic polynomials and with random coefficients $c \in [-10, 10]$ of floating-point numbers. For noise, we have generated a pair of polynomials $F_{\mathrm{N}}(x)$ and $G_{\mathrm{N}}(x)$ of degrees $m-1$ and $n-1$, respectively, with random coefficients as the same as for $F_0(x)$ and $G_0(x)$. Then, we have defined a pair of test polynomials $F(x)$ and $G(x)$ as

$$F(x) = F_0(x) + \frac{e_F}{\|F_{\mathrm{N}}(x)\|_2} F_{\mathrm{N}}(x), \quad G(x) = G_0(x) + \frac{e_G}{\|G_{\mathrm{N}}(x)\|_2} G_{\mathrm{N}}(x),$$

respectively, scaling the noise such that the 2-norm of the noise for F and G is equal to e_F and e_G, respectively. In the present test, we set $e_F = e_G = 0.1$.

In this test, we have compared our implementation against a method based on the structured total least norm (STLN) method [7], using their implementation (see Acknowledgments). In their STLN-based method, we have used the procedure `C_con_mulpoly` which calculates the approximate GCD of several polynomials in $\mathbf{C}[x]$. The tests have been carried out on Maple 12 with `Digits=15` executing hardware floating-point arithmetic. For every example, we have generated 100 random test polynomials as in the above. In executing a modified Newton method, we set a threshold of the 2-norm of the "update" vector in each iteration $\varepsilon = 1.0 \times 10^{-8}$; in `C_con_mulpoly`, we set the tolerance $e = 1.0 \times 10^{-8}$.

Table 1 shows the results of the test: m and n denotes the degree of a pair F and G, respectively, and d denotes the degree of approximate GCD. The columns with "STLN" are the data for the STLN-based method, while those with "GPGCD" are the data for the GPGCD method. "Error" is the perturbation $\|\tilde{F} - F\|_2^2 + \|\tilde{G} - G\|_2^2$, where "$ae{-}b$" denotes $a \times 10^{-b}$; "#Iterations" is the number of iterations; "Time" is computing time in seconds.

We see that, in the most of tests, both methods calculate approximate GCD with almost the same amount of perturbations, while the GPGCD method runs much faster than STLN-based method by approximately from 10 to 30 times. Note that, in contrast to the real coefficient case [17], both methods have converged in all the test cases with the number of iterations and sufficiently small amount of perturbations as approximately equal to those shown as in Table 1.

5 Concluding Remarks

Based on our previous research [17], we have extended our GPGCD method for polynomials with the complex coefficients.

Our experiments have shown that, as in the real coefficients case [17], our algorithm calculates approximate GCD with perturbations as small as those calculated by methods

Ex.	m, n	d	Error		#Iterations		Time (sec.)	
			STLN	GPGCD	STLN	GPGCD	STLN	GPGCD
1	10, 10	5	$3.72e{-3}$	$3.72e{-3}$	4.48	4.43	1.79	0.15
2	20, 20	10	$4.16e{-3}$	$4.16e{-3}$	4.24	4.22	5.88	0.30
3	30, 30	15	$4.33e{-3}$	$4.33e{-3}$	4.54	4.48	14.29	0.58
4	40, 40	20	$4.48e{-3}$	$4.48e{-3}$	4.08	4.08	24.10	0.88
5	50, 50	25	$4.63e{-3}$	$4.64e{-3}$	4.05	4.12	39.19	1.36
6	60, 60	30	$4.61e{-3}$	$4.61e{-3}$	4.02	4.06	60.48	1.96
7	70, 70	35	$4.82e{-3}$	$4.82e{-3}$	3.90	4.02	84.51	2.66
8	80, 80	40	$4.84e{-3}$	$4.84e{-3}$	3.88	4.04	116.03	3.65
9	90, 90	45	$4.79e{-3}$	$4.79e{-3}$	3.85	4.01	151.27	4.66
10	100, 100	50	$4.77e{-3}$	$4.78e{-3}$	3.83	4.06	199.48	6.00

Table 1: Test results for large sets of polynomials with approximate GCD. See Section 4 for details.

based on the structured total least norm (STLN) method, while our method has shown significantly better performance over the STLN-based methods in its speed, by approximately up to 30 times, which seems to be sufficiently practical for inputs of low or moderate degrees. This result shows that, in contrast to their *structure preserving* method, our simple method can achieve accurate and efficient computation as or more than theirs in calculating approximate GCDs.

Our future research includes theoretical investigation of convergence properties, investigation for efficient computation in solving a linear system in each iteration by analysis of the structure of matrices, generalization of our method to several input polynomials, and so on.

Acknowledgments

We thank Professor Erich Kaltofen for making their implementation for computing approximate GCD available on the Internet and providing experimental results. We also thank the anonymous reviewers for their helpful comments.

This research was supported in part by the Ministry of Education, Culture, Sports, Science and Technology of Japan, under Grant-in-Aid for Scientific Research (KAKENHI) 19700004.

References and Notes

[1] B. Beckermann and G. Labahn. A fast and numerically stable Euclidean-like algorithm for detecting relatively prime numerical polynomials. *J. Symbolic Comput.*, 26(6):691–714, 1998.

[2] P. Chin, R. M. Corless, and G. F. Corliss. Optimization strategies for the approximate GCD problem. In *Proc. ISSAC'98*, pages 228–235. ACM, 1998.

[3] R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt. The singular value decomposition for polynomial systems. In *Proc. ISSAC'95*, pages 195–207. ACM, 1995.

[4] R. M. Corless, S. M. Watt, and L. Zhi. QR factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Process.*, 52(12):3394–3402, 2004.

[5] J. W. Demmel. *Applied numerical linear algebra*. SIAM, 1997.

[6] I. Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate GCDs. *J. Pure Appl. Algebra*, 117/118:229–251, 1997.

[7] E. Kaltofen, Z. Yang, and L. Zhi. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *Proc. ISSAC'06*, pages 169–176. ACM, 2006.

[8] E. Kaltofen, Z. Yang, and L. Zhi. Structured low rank approximation of a Sylvester matrix. In D. Wang and L. Zhi, editors, *Symbolic-Numeric Computation*, pages 69–83. Birkhäuser, 2007.

[9] N. K. Karmarkar and Y. N. Lakshman. On approximate GCDs of univariate polynomials. *J. Symbolic Comput.*, 26(6):653–666, 1998.

[10] N. Ohsako, H. Sugiura, and T. Torii. A stable extended algorithm for generating polynomial remainder sequence (in Japanese). *Trans. Japan Soc. Indus. Appl. Math*, 7(3):227–255, 1997.

[11] V. Y. Pan. Computation of approximate polynomial GCDs and an extension. *Inform. and Comput.*, 167(2):71–85, 2001.

[12] J. B. Rosen. The gradient projection method for nonlinear programming. II. Nonlinear constraints. *J. Soc. Indust. Appl. Math.*, 9:514–532, 1961.

[13] M. Sanuki and T. Sasaki. Computing approximate GCDs in ill-conditioned cases. In *Proc. SNC'07*, pages 170–179. ACM, 2007.

[14] T. Sasaki and M.-T. Noda. Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations. *J. Inform. Process.*, 12(2):159–168, 1989.

[15] A. Schönhage. Quasi-gcd computations. *J. Complexity*, 1(1):118–137, 1985.

[16] K. Tanabe. A geometric method in nonlinear programming. *J. Optim. Theory Appl.*, 30(2):181–210, 1980.

[17] A. Terui. An iterative method for computing approximate GCD of univariate polynomials. In *Proc. ISSAC'09*, pages 351–358. ACM, 2009.

[18] C. J. Zarowski, X. Ma, and F. W. Fairman. QR-factorization method for computing the greatest common divisor of polynomials with inexact coefficients. *IEEE Trans. Signal Process.*, 48(11):3042–3051, 2000.

[19] Z. Zeng. The approximate GCD of inexact polynomials, Part I: a univariate algorithm (extended abstract). preprint. 8 pages.

[20] L. Zhi. Displacement structure in computing approximate GCD of univariate polynomials. In *Proc. the Sixth Asian Symposium on Computer Mathematics (ASCM 2003)*, pages 288–298. World Scientific, 2003.

Towards the calculation of Casimir forces for inhomogeneous planar media

C. Xiong[1], T.W. Kelsey[1*], S.A. Linton[1] and U. Leonhardt[2]

[1]School of Computer Science, University of St Andrews, KY16 9SX, UK
[xc,tom,sal]@cs.st-andrews.ac.uk

[2] School of Physics and Astronomy, University of St Andrews, KY16 9SS, UK
ulf@st-andrews.ac.uk

.....

Abstract

Casimir forces arise from vacuum fluctuations. They are fully understood only for simple models, and are important in nano- and microtechnologies. We report our experience of computer algebra calculations towards the Casimir force for models involving inhomogeneous dielectrics. We describe a methodology that greatly increases confidence in any results obtained, and use this methodology to demonstrate that the analytic derivation of scalar Green's functions is at the boundatry of current computer algebra technology. We further demonstrate that Lifshitz theory of electromagnetic vacuum energy can not be directly applied to calculate the Casimir stress for models of this type, and produce results that indicate the possibility of alternative regularisations. We discuss the relative strengths and weaknesses of computer algebra systems when applied to this type of problem, and suggest combined numerical and symbolic approaches towards a more general computational framework.

1 Introduction

Casimir forces result from zero-point vacuum fluctuations confined between two dielectric materials [4]. Although these forces were predicted theoretically in the 1940s, empirical evidence confirming the theory has only been obtained in recent years [3, 5, 7, 9, 11]. Casimir forces are important in nanotechnology and microtechnology: repulsive Casimir forces can reduce friction in nano- and micromechanical devices, whereas attractive forces can "glue" components together that are designed to be free-moving. Lifshitz theory [6] is a theoretical approach to the calculation of Casimir forces, in which the Green's tensor for the electric field is used to derive electromagnetic stress and energy density.

The standard planar model is to have two plates, L and R, of uncharged dielectric materials separated in the x direction by a few micrometers. The materials have permittivities $\varepsilon_L(x, i\xi)$ and $\varepsilon_R(x, i\xi)$, depending on displacement and frequency ξ, which completely describe the media since we assume that there is no magnetic response (we enforce $\mu_L(x, i\xi) = 1 = \mu_R(x, i\xi)$ for the magnetic permeabilities involved). The gap between the plates, C, is either a quantum vacuum or third dielectric with $\varepsilon_C(x, i\xi)$ equal to a constant; we consider such a model to be *homogeneous*. For this model, and for variations of this model that include moving plates [12], the Casimir force can be both calculated analytically and measured empirically [1]. For extensions of this model involving more than two plates, numerical methods can be used to obtain the Casimir forces for specific types of plate [10].

*Correspondence to: Dr Tom Kelsey, School of Computer Science, University of St Andrews, KY16 9SX, UK; +44 (0)1334 463249 or +44 (0)1334 463253

In this paper we consider *inhomogeneous* models where the permittivity of the central region, $\varepsilon_C(x, i\xi)$, varies with x. The primary aim of the paper is to see which, if any, inhomogeneous models allow the analytic derivation of their Casimir forces from calculations performed in the widely-used computer algebra systems Maple (Waterloo Maple Inc., London, Ontario, Canada) and Mathematica (Wolfram Research Inc., Champaign, IL, USA). All computations were run on an Intel Xeon E5430 2.66GHz with 8 cores and 16 Gb memory, using Maple version 13.0 and Mathematica version 7.0 under CentOS 5. In particular we explore the applicability of Lifshitz theory to inhomogeneous models. The Lifshitz regularisation process described in Section 2 was derived with homogeneous media in mind, we therefore explore the possibility that this could be a confounding factor in attempts to calculate Casimir forces in the inhomogeneous case.

In Section 2 we give the standard Lifshitz theoretic approach to deriving Casimir stresses for homogeneous media, and discuss how these may be calculated using Maple and Mathematica. In Section 3 we describe our methodology for performing and checking similar calculations for inhomogeneous models, and outline the strengths and weakness of the two computer algebra systems. We present results that suggest that many, but not all, inhomogeneous models can not be dealt with analytically using current computer algebra capabilities. Section 4 contains our analysis of standard Lifshitz theory applied to the model in which the central permittivity decays exponentially ($\varepsilon_C(x, i\xi) = ae^{-bx}$), together with results that suggest that suitable alternative regularisations may be deriveable. In Section 5 we discuss computational aspects, such as the limitations of existing computer algebra systems, and the possibility of future numeric-symbolic approaches.

2 The calculation of Casimir stress in planar media

In this section we describe the mathematical and physical concepts involved in our computations, and present the sequence of calculations involved in determining the Casimir force for planar models. A more detailed exposition of the underlying physics, together with full derivation of the equations involved and descriptions of theoretic approaches other than that of Lifshitz, is given in [1]. The resulting sequence of calculations can, in principle, be done by hand, using symbolic computer algebra, via numeric techniques, or by a combined numeric-symbolic approach. We report on our experiences of the second of these options in Section 3.

Stresses on objects in electromagnetic fields are given by Maxwell's stress tensor, in which $\hat{\mathbf{E}}$ and $\hat{\mathbf{H}}$ are respectively the electric and magnetic fields, $\hat{\mathbf{B}}$ is the magnetic induction and $\hat{\mathbf{D}}$ is the electric displacement.

$$\hat{\sigma} = \hat{\mathbf{E}} \otimes \hat{\mathbf{D}} + \hat{\mathbf{B}} \otimes \hat{\mathbf{H}} - \frac{1}{2}(\hat{\mathbf{E}} \cdot \hat{\mathbf{D}} + \hat{\mathbf{B}} \cdot \hat{\mathbf{H}})\mathbb{I}_3 \tag{1}$$

For stationary electromagnetic fields, the divergence of the Maxwell's stress tensor gives the force density $\hat{\mathbf{f}}$,

$$\hat{\mathbf{f}} = \nabla \cdot \hat{\sigma}. \tag{2}$$

The expectation values (also known as correlation functions) for the tensor products in Equation (1) are related to the retarded Green's function as follows:

$$\langle \hat{\mathbf{E}}(\mathbf{r}, t) \otimes \hat{\mathbf{D}}(\mathbf{r}', t) \rangle = -\frac{\hbar}{\pi c^2} \int_0^\infty d\xi \varepsilon(\mathbf{r}, i\xi) \xi^2 \mathbf{G}(\mathbf{r}, \mathbf{r}', i\xi), \tag{3}$$

$$\langle \hat{\mathbf{B}}(\mathbf{r},t) \otimes \hat{\mathbf{H}}(\mathbf{r}',t) \rangle = \frac{\hbar}{\pi} \int_0^\infty d\xi \frac{1}{\mu(\mathbf{r}, i\xi)} \nabla \times \mathbf{G}(\mathbf{r}, \mathbf{r}', i\xi) \times \overleftarrow{\nabla'} \tag{4}$$

The notation $\times \overleftarrow{\nabla'}$ denotes a curl on $\mathbf{r}'$ in $\mathbf{G}(\mathbf{r}, \mathbf{r}', i\xi)$ from the right. $\mathbf{G}(\mathbf{r}, \mathbf{r}', i\xi)$ is the retarded Green's tensor for the vector potential in a Coulomb gauge, and is defined as the solution of the following inhomogeneous electromagnetic wave equation

$$\nabla \times \frac{1}{\mu} \nabla \times \mathbf{G}(\mathbf{r}, \mathbf{r}', i\xi) + \varepsilon \frac{\xi^2}{c^2} \mathbf{G}(\mathbf{r}, \mathbf{r}', i\xi) = \delta(\mathbf{r} - \mathbf{r}') \mathbb{I}_3. \tag{5}$$

The Green's function should always obey the reciprocity relation:

$$\mathbf{G}(\mathbf{r}, \mathbf{r}', i\xi) = \mathbf{G}(\mathbf{r}', \mathbf{r}, -i\xi); \tag{6}$$

we describe our extensive use of this as a check for correctness of our calculated scalar Green's functions in Section 3.
We are considering planar dielectrics, for which the permittivity $\varepsilon(\mathbf{r}, i\xi) = \varepsilon(x, i\xi)$ and magnetic permeability $\mu(\mathbf{r}, i\xi) = \mu(x, i\xi)$, i.e. depend only on the x-coordinate. The Green's function in terms of its Fourier transform in y and z is

$$\mathbf{G}(x, x', u, v, i\xi) = \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dz \mathbf{G}(\mathbf{r}, \mathbf{r}', i\xi) e^{-iu(y-y') - iv(z-z')}. \tag{7}$$

The Fourier-transformed Green's function $\mathbf{G}(x, x', u, v, i\xi)$ is given by the Fourier-transformed wave equation:

$$\nabla \times \frac{1}{\mu(x, i\xi)} \nabla \times \mathbf{G}(x, x', u, v, i\xi) + \varepsilon(x, i\xi) \frac{\xi^2}{c^2} \mathbf{G}(x, x', u, v, i\xi) = \delta(x - x'). \tag{8}$$

The Casimir force depends only on the $xx-$component of Maxwell's stress tensor because the force density is also independent of y and z. In the limit $\mathbf{r} \to \mathbf{r}'$, the result for σ_{xx} is

$$\begin{aligned} \sigma_{xx} &= -\frac{\hbar c}{8\pi^3} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_0^{\infty} u \left(\frac{1}{\mu} (w^2 - \partial_x \partial_{x'}) \tilde{g}_{Es} + \frac{1}{\varepsilon} (w^2 - \partial_x \partial_{x'}) \tilde{g}_{Ms} \right) du dv d\kappa \mid_{\mathbf{x}'=\mathbf{x}} \\ &= -\frac{\hbar c}{4\pi^2} \int_0^{\infty} du \int_0^{\infty} d\kappa u \left(\frac{1}{\mu} (w^2 - \partial_x \partial_{x'}) \tilde{g}_{Es} + \frac{1}{\varepsilon} (w^2 - \partial_x \partial_{x'}) \tilde{g}_{Ms} \right) \mid_{v=0, \mathbf{x}'=\mathbf{x}}, \end{aligned} \tag{9}$$

with

$$\begin{aligned} w &= \sqrt{u^2 + v^2 + \varepsilon \mu \kappa^2}, \quad \kappa = \frac{\xi}{c}, \\ \tilde{g}_{Es} &= \tilde{g}_E - \mu \tilde{g}_0, \quad \text{and} \quad \tilde{g}_{Ms} = \tilde{g}_M - \varepsilon \tilde{g}_0. \end{aligned}$$

In Lifshitz theory, $\tilde{g}_{Es}$ and $\tilde{g}_{Ms}$ are the regularized electric and magnetic Green's functions (where regularisation involves subtraction of the the relevant divergent part). $\tilde{g}_0$ is the infinite contribution from the retarded Green's function in a space with homogeneous medium:

$$\tilde{g}_0 = -\frac{1}{2w} e^{(-w|x-x'|)}. \tag{10}$$

In summary, to obtain the Casimir force for a planar dielectric model, the sequence of calculations is:

1. calculate the scalar Green's functions – Equation (8) – for the permittivity of the specific media under consideration (recalling the modelling assumption $\mu(x, i\xi) = 1$ described in Section 1)

2. perform the regularisation that removes the infinite parts from the above scalar Green's functions

3. solve the double integral – Equation (9) – to obtain the stress tensor σ_{xx}

4. the divergence of σ_{xx} is the theoretically predicted Casimir force – Equation (2).

In the homogeneous case all the calculations can be performed analytically using either Maple or Mathematica, since $\varepsilon(x, i\xi)$ does not vary with x. Equation (8) reduces to

$$\frac{d^2}{dx^2}\tilde{g}(x) - (u^2 + v^2 + \varepsilon\kappa^2)\tilde{g}(x) = \delta(x - x') \tag{11}$$

where $\tilde{g}$ denotes either the electric or magnetic Green's function, and in which none of the left-hand parameters depends on x. The general solution involves trigonometric functions and the Heaviside function; specific solutions are easily obtained from the boundary conditions. The subtractions involved in stage 2 are also straightforward, again since neither μ nor ε varies with x. The double integrals with infinite ranges produce finite results, since the integrand converges to zero with increasing u and κ. Stage 4 is relatively simple. In Section 3 we commence our analysis of how the computational details are affected when homogeneity is no longer assured.

3 Specific inhomogeneous permittivity models

For a given model of the permittivity $\varepsilon(x)$ and $\mu = 1$ (no magnetic response), we first need to find the scalar Green functions,

$$\frac{d^2\tilde{g}_E}{dx^2} - \left(u^2 + \varepsilon(x)\kappa^2\right)\tilde{g}_E = \delta(x - x'), \tag{12}$$

$$\frac{d}{dx}\left(\frac{1}{\varepsilon(x)}\frac{d}{dx}\tilde{g}_M\right) - \left(\frac{u^2}{\varepsilon(x)} + \kappa^2\right)\tilde{g}_M = \delta(x - x'). \tag{13}$$

An intrinsic problem is the assessment of the validity of any results. The output from a computer algebra system will consist of symbolic expressions, which can be evaluated as real numbers for supplied values of the parameters involved. Empirical validation is not known to be possible for all models, and is expensive, time-consuming and technically demanding.

Our solution is to constrain our results to be the same when solving from the left and from the right. This is not a guarantee that any results obtained are correct, but it does increase confidence and allows us to detect models for which analytic solution is intractable using current computer algebra capabilities. Our methodology therefore is to perform the calculations using two computer algebra systems (Maple and Mathematica) and to use the reciprocity relations given in Equation (6). If the results from the two independent systems coincide, and if the results are the same from either the left 'or from the right, then we have a high level of confidence in their correctness.

For models such as $\varepsilon(x) = 1 + e^{-x}$ and $\varepsilon(x) = x^2$ using Maple we find that the reciprocity relations are violated in the magnetic case. Using Mathematica to perform the

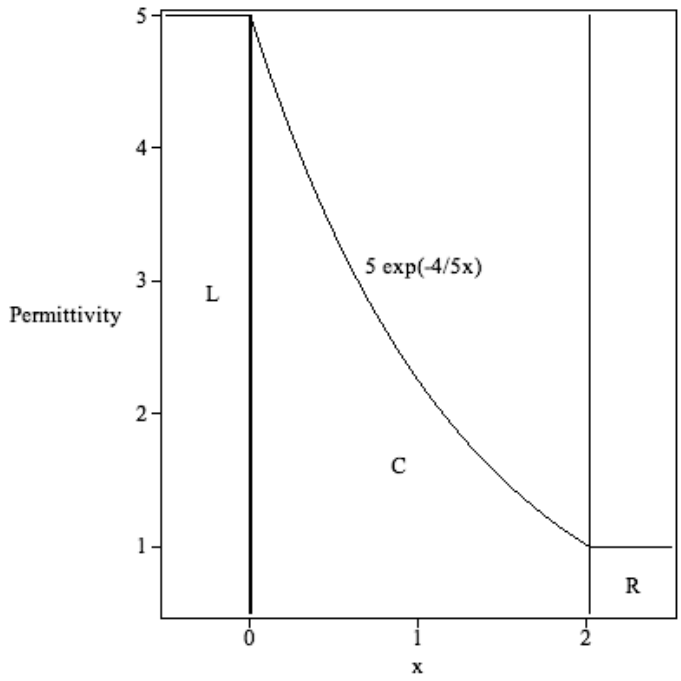


Figure 1: The inhomogeneous model given by $\varepsilon_L = 5$, $\varepsilon_C = 5\exp(-4/5x)$ and $\varepsilon_R = 1$. The central part runs from $x = 0$ to $x = log(a)/b \approx 2.0118$.

same calculations, we were unable to solve the equations analytically, and were therefore unable to perform the reciprocity check. The difference between the two systems is that recent versions of Maple contain an implementation of the use of Heun's functions [8] to solve ODEs. Heun's functions are the solutions of the Heun form of 2nd order linear ODEs; any such ODE can be converted into Heun form. Equation (12) is a linear 2nd order ODE, so, in Maple, it can be converted to Heun form and solved directly, with perfect agreement from the right and left directions. Equation (13) is non-linear, however. We can convert it into a modified Heun ODE and obtain a solution in the form of the product of a Heun C function and an exponential correction factor. There is a discrepancy between the left and right solutions, which appears to be introduced when the correction factor is computed. Unfortunately, we are never sure which, if either, of these results is correct, hence further work is needed to correct the Maple implementation. Mathematica, on the other hand, has no Heun ODE or function capabilities, and neither the linear nor nonlinear ODEs could be solved analytically.

We have found only one permittivity model for which the scalar Green's functions can be derived analytically (i.e. without using any numeric calculation options) with the reciprocity relations fully satisfied (Figure 1 is an illustrative example). This is exponentially decaying permittivity,

$$\varepsilon(x) = ae^{-bx}, \qquad \text{for positive constants } a \text{ and } b, \tag{14}$$

bounded on each side by homogeneous dielectrics.

For this case, the general solutions for $\tilde{g}_E$ and $\tilde{g}_M$ were found to be

$$\begin{aligned}\tilde{g}_E &= C_{E1}I_{\nu 1}(-\lambda) + C_{E2}K_{\nu 1}(\lambda) + \frac{2}{b}(I_{\nu 1}(-\lambda)K_{\nu 1}(\lambda') \\ &\quad -I_{\nu 1}(-\lambda')K_{\nu 1}(\lambda))\text{Heaviside}(\lambda-\lambda'), \end{aligned} \tag{15}$$

$$\begin{aligned}\tilde{g}_M &= C_{M1}\lambda I_{\nu 2}(-\lambda) + C_{M2}\lambda K_{\nu 2}(\lambda) \\ &\quad + \frac{b\lambda\lambda'}{2\kappa^2}(I_{\nu 2}(-\lambda)K_{\nu 2}(\lambda') - I_{\nu 2}(-\lambda')K_{\nu 2}(\lambda))\text{Heaviside}(\lambda-\lambda'), \end{aligned} \tag{16}$$

$$\text{where} \quad \lambda = \frac{2\kappa\sqrt{a}}{b}e^{-bx/2}, \quad \nu 1 = 2u^2/b, \quad \nu 2 = \sqrt{1+\nu 1^2}, \tag{17}$$

and in which the Cs are arbitrary coefficients determined by the continuity of

$$\tilde{g}_E, \quad \tilde{g}_M, \quad \frac{1}{\mu(x,i\kappa)}\partial_x\tilde{g}_E, \quad \text{and } \frac{1}{\varepsilon(x,i\kappa)}\partial_x\tilde{g}_M \tag{18}$$

at the boundaries, and I and K are the modified Bessel functions.

Two interesting computational aspects were encountered. Firstly, Mathematica does not return solutions of the PDEs in terms of Bessel K function; it instead returns expressions involving Gamma functions which are mathematically equivalent, but which are lengthy and hard for humans to interpret. Secondly, intermediate Maple output suggested the variable changes involving λ, $\nu 1$ and $\nu 2$ – Equations (17). These simplifying re-arrangements both (i) greatly aid the efficiency of the remaining calculations, and (ii) helped us to interpret and check the results. The calculations were therefore easier to perform in Maple than in Mathematica, but, for this model, both systems returned the same results, from the left and from the right, when evaluated as floats. We are therefore confident that our scalar Green's functions are exactly those needed for the Lifshitz regularisation process.

4 The testing of standard Lifshitz regularisation

Standard Lifshitz theory involves the subtraction of the contribution to the stress that does not arise from material inhomogeneity. This is known to produce accurate (i.e. empirically verifiable) results for the standard model where the gap between the two plates is either empty or filled with a homogeneous dielectric. However, this approach is known to result in an infinite Casimir force in models involving cylinders and spheres [2]. In certain cases an alternative regularisation has been found (but not always agreed upon by the expert community), whilst for others the problem of calculating a finite stress using any theoretical approach remains unsolved. For our model (exponentially decaying permittivity for the central medium) we therefore expect to either (i) use standard Lifshitz theory to calculate a finite Casimir force, (ii) derive an infinite force, with the structure of the results indicating the possibility of alternative regularisation, or (iii) derive an infinite force, with no clues on how to proceed.

Our results, displayed for illustrative parameter choices in Figure 2, indicate that the for one of the wave parameters (κ) we obtain convergence to a finite integrand, but for the other (u) the integrand diverges. Unfortunately, we can no longer consider κ and u to be the respective x and y wave components, since we have performed a Fourier transformation.

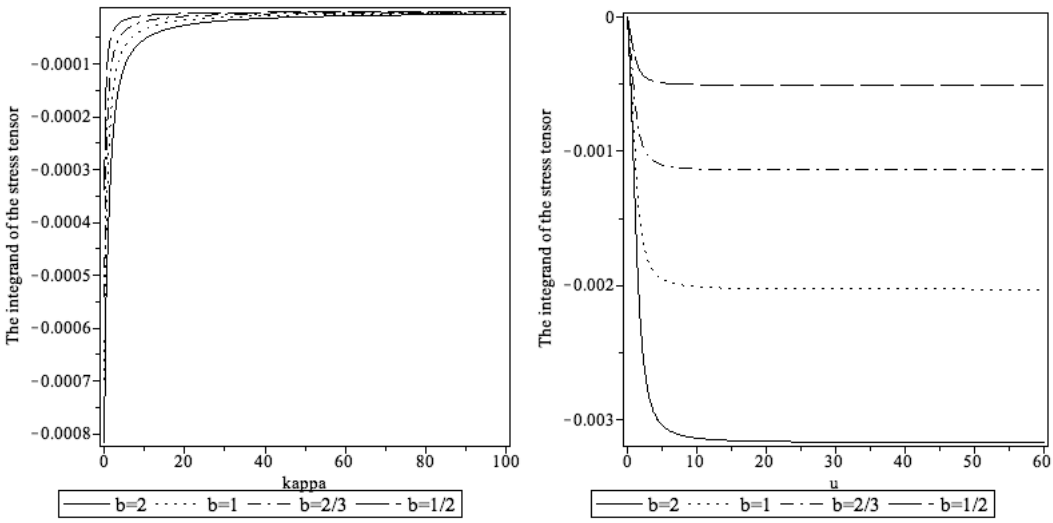


Figure 2: The integrand of the stress tensor obtained from Green's functions regularised using Lifshitz theory. We use the conventions that the physical constants $\hbar = c = 1$. The central permittivity model is $\varepsilon(x) = 5\exp(-bx)$. As κ increases, the integrand converges to zero (left plot). As u increases, the integrand converges to a nonzero constant, the value of which depends on the model parameter b but not on a (right plot).

Further investigation of the divergence shows that the constant nonzero value depends neither on a or x, but only on b (Figure 3). Routine simplification, (setting $x = x'$) gives the divergence constant DC for the stress tensor as a function of b:

$$DC \approx -0.0036b^2, \tag{19}$$

This divergent behaviour leads to the prediction of an infinite Casimir force, which is not a physically realistic outcome. However, the predictability of the amount of divergence suggests that it may be possible to modify Lifshitz theory for this model, so that a plausible finite Casimir force is predicted. Such as regularisation has recently been proposed [13], with the resulting Casimir forces calculated using the methodology presented in this paper.

5 Conclusions

Our findings suggest that the calculation of scalar Green's functions for arbitrary inhomogeneous media is at the boundary of the current capabilities of Maple and Mathematica. Using Maple we can get satisfactory results for exactly one model, and believe we could increase the number of such models if the modified Heun function implementation within Maple were to be improved. Mathematica is less useful for these calculations, as no Heun function implementation is present in the current system. However, we have successfully replicated Maple results using Mathematica, indicating that the lengthy and complex Mathematica expressions produced as intermediate output are completely correct.

We are highly confident that our scalar Green's function calculations are accurate. In addition to the approach described in Section 3, we split the Green's functions into bare and

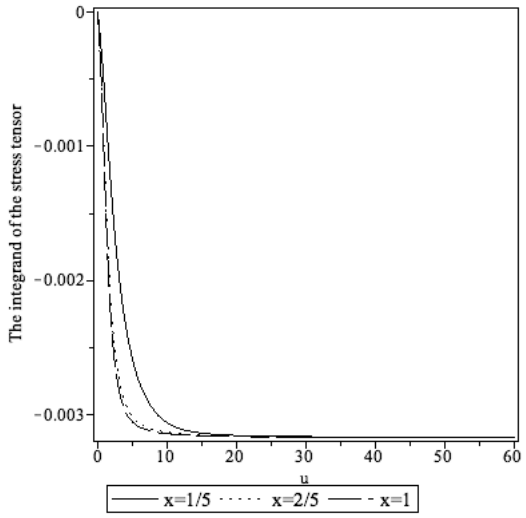


Figure 3: The integrand of the stress tensor obtained from Green's functions regularised using Lifshitz theory. We use the conventions that the physical constants $\hbar = c = 1$. The central permittivity model is $\varepsilon(x) = 5\exp(-bx)$. Model parameters have been set as $a = 5$ and $b = 4/5$. We observe that for fixed b, the nonzero convergence value is the same for all choices of x in the central region.

scattered parts, allowing us to derive the specific solution of the bare part using initial rather than boundary conditions. The results of these calculations agree with those described in this paper for both systems, and hence also satisfy out reciprocity constraints.

In Sections 3 and 4 we discussed the first two stages of Casimir force prediction using Lifshitz theory. The third stage, a complicated double integration over infinite ranges, is, in general, not computable analytically in either Maple or Mathematica for inhomogeneous models. Instead, we substitute a realistically high finite value for the infinities and obtain numeric approximations. For example, in the model presented in Figures 2 and 3 with $u = 150$ the integrand has a magnitude of 10^{-13}, descreasing to zero with increasing u. Stage 4 is well within the capabilities of any decent computer algebra system.

Future avenues of research include (i) the testing of any proposed alternative regularisation using our methodology of comparing results from two systems for both the the right and left limits, (ii) the development of a combined numeric-symbolic framework that agrees with the symbolically derived results presented here for the exponential model, and which can be used to calculate Casimir forces for those inhomogeneous model that lie beyond the current analytic capabilities of Maple and Mathematica.

Acknowledgements

The authors are supported by EPSRC grant EP/CS23229/1. UL is also supported by a Royal Society Wolfson Research Merit Award. We would like to acknowledge the critical discussions we have had with Tom Philbin.

References and Notes

[1] M. Bordag, G. L. Klimchitskaya, U. Mohideen, and V. M. Mostepanenko. *Advances in the Casimir Effect*. Oxford University Press, Oxford, 2009.

[2] T. H. Boyer. Quantum electromagnetic zero-point energy of a conducting spherical shell and the Casimir model for a charged particle. *Physical Review*, 174(5):1764–1776, Oct 1968.

[3] G. Bressi, G. Carugno, R. Onofrio, and G. Ruoso. Measurement of the Casimir force between parallel metallic surfaces. *Physical Review Letters*, 88(4), Jan 2002.

[4] H B G Casimir. On the attraction between two perfectly conducting plates. *Proc. K. Ned. Akad. Wet.*, 51:793–795, 1948.

[5] R. S. Decca, D. López, E. Fischbach, and D. E. Krause. Measurement of the Casimir force between dissimilar metals. *Physical Review Letters*, 91(5), Jul 2003.

[6] I. E. Dzyaloshinskii, E. M. Lifshitz, and L. P. Pitaevskii. The general theory of van der Waals forces. *Advances in Physics*, 10(38):165–209, April 1961.

[7] C. Hertlein, L. Helden, A. Gambassi, S. Dietrich, and C. Bechinger. Direct measurement of critical Casimir forces. *Nature*, 451(7175):172–175, January 2008.

[8] K Heun. Theorie der Riemann'schen Functionen zweiter Ordnung mit vier Verzweigungspunkten. *Mathematische Annalen*, 3:161, 1899.

[9] S. K. Lamoreaux. Demonstration of the Casimir force in the 0.6 to 6 μm range. *Physical Review Letters*, 78(1):5–8, January 1997.

[10] L. D. Landau, E. M. Lifshitz, and L. P. Pitaevskii. *Statistical Physics, Part 2*. Butterworth-heinemann, Oxford, 1980.

[11] J. N. Munday, F. Capasso, and A. V. Parsegian. Measured long-range repulsive Casimir–Lifshitz forces. *Nature*, 457(7226):170–173, 2009.

[12] T. G. Philbin and U. Leonhardt. No quantum friction between uniformly moving plates. *New Journal of Physics*, 11(3):033–035, 2009.

[13] T.G. Philbin, C. Xiong, and U. Leonhardt. Casimir stress in an inhomogeneous medium. *arXiv:0909.2998*, 2009. Preprint.

ASCM Organized Session

Digitizing Mathematics

Validated Numerical Computation

Computational Algebraic Number Theory

Digitized Mathematical Literature and the Semantic Web Invited Talk

David Ruddy

Director, E-Publishing Techologies
Cornell University Library,
Ithaca, New York, USA
dwr4@cornell.edu

Abstract

Project Euclid and other repositories have assembled many millions of pages of digitized mathematical literature. Our goal has been to make mathematical literature, much of it published initially on paper over the last century, accessible via the World Wide Web. This content, however, is relatively underutilized, since access has been primarily conceived as a human activity — discovering and viewing particular works of mathematical literature. We discuss here principles of the Semantic Web and best practices from the Linked Data effort, and how they could be relatively easily implemented within the mathematical community and large document repositories. Such efforts would allow computers access to a large and rich store of data, with the ability to infer relationships among documents. New user services could then be layered on top of these capabilities.

Extract Baseline Information Using Support Vector Machine

Walaa ALY[1], Seiichi UCHIDA[1*], Masakazu SUZUKI[2]

[1] Department of Intelligent Systems, Kyushu University,
744 Motooka, Nishi-ku, Fukuoka-shi, 819-0395 Japan
walaa@human.is.kyushu-u.ac.jp, uchida@human.is.kyushu-u.ac.jp

[2] Department of Mathematics, Kyushu University,
6-10-1, Hakozaki, Higashi-ku, Fukuoka-shi, 812-8581 Japan
suzuki@math.kyushu-u.ac.jp

Abstract

Printed mathematical documents have many features which differ from text documents. These features include two-dimensional structure, such as subscripts, superscripts, and scalable symbols, such as minus, summation. Due to these features, the recognition of printed mathematical documents becomes very challenging. In this paper, a support vector machine (SVM) classifier is used to extract baseline information, which provides a basis for finding the two-dimensional structure from printed mathematical documents. Experiments using very large databases show high efficiency of the proposed method with the extraction accuracy reaching 99.25 %.

1 Introduction

Mathematical expressions are considered as an essential part in any scientific and technical document. Recognition of mathematical expressions have two major steps. (i) Recognition of symbols: each symbol is recognized using an OCR approach. (ii) Structure analysis: from the set of recognized symbols and their bounding boxes, the expression is reconstructed by analyzing the two-dimensional structure of the expression. A survey of past attempts on mathematical expression recognition can be found in [1].

In this paper, we will mainly focus on the structure analysis step of mathematical expressions. Many researchers have discussed the structure analysis step, starting with Anderson [2], where a purely syntactic approach was introduced for parsing mathematical expressions. Okamoto and his colleagues [3] used geometric information to find the structure of the expression. Twaakyondo et al. [4] used two strategies, namely, top-down and bottom-up, to determine the structure of the expression.

In past attempts, the evaluations have been carried out with respect to the total performance of the system and, therefore, they have not evaluated the individual steps of the structure analysis. That is, those past attempts gave neither qualitative nor quantitative analysis of the discrimination task [‡]. In contrast, this paper gives a solid ground to the structure analysis step through several evaluations with very large databases.

In this paper, baseline information is extracted as a classification task for mathematical expressions. Figure 1 illustrates the classification task. *Baseline* is a line containing only symbols which are not vertically offset from other symbols. Hereafter, these symbols will be called baseline symbols. For example, the expression "$A_2 + B_1$" has baseline symbols "A, ""+, " and "B." In contrast, *non-baseline* is a line containing symbols which vertically offset from baseline symbols. Hereafter, these symbols will be called non-baseline symbols. For example, the expression "$X^2 + Y_5$" has

*Correspondence to: 744 Motooka, Nishi-ku, Fukuoka-shi, 819-0395 Japan and Tel:092-802-3574, Fax:092-802-3600.

‡Moreover, in other attempts [5, 6, 7, 8, 9, 10, 11, 12, 13, 14], the detail of the structure part is completely concealed as a black box of a large math OCR system.

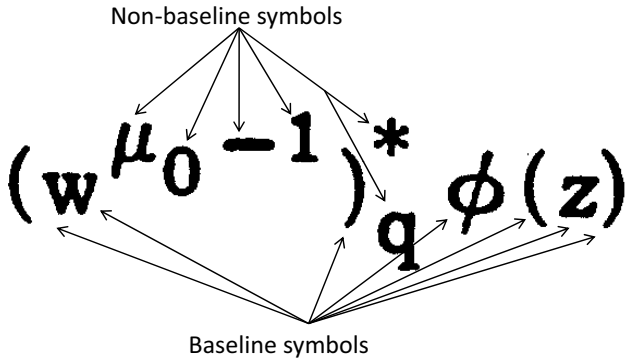


Figure 1: Baseline/non-baseline classification problem.

$$(h^{\mu}_{ik} h^{\mu}_{jl} - h^{\mu}_{il} h^{\mu}_{jk})$$

$$\sum c_{ij} g_k^{q_k - i} g_{k-1}^{q_{k-1} - j}$$

$$\mathrm{H}^r(\mathrm{K}, \mathrm{A}) \times \mathrm{H}^{1-r}(\mathrm{K}, \hat{\mathrm{A}})$$

Figure 2: Examples of mathematical expressions.

baseline symbols "X," "+," and "Y," and non-baseline symbols "2," and "5." We will classify each symbol into baseline and non-baseline as the most important step of structure analysis. In the following, all mathematical symbols, such as "$*$," "$\cup$," and "$\int$," and alphanumeric characters, such as "A," "2," and "η," will be called symbols.

The classification task is not trivial. Figure 2 depicts several superscripts and subscripts in various mathematical expressions. In the bottom example, it is difficult to distinguish non-baseline characters "r" and "$1 - r$" from the baseline characters (e.g., "H") by using only the offset. They clearly show how difficult it is to extract baseline information.

Although baseline information is very important, there is only little related work, to the authors' best knowledge. Zanibbi et al [15] is related to our work. They proposed a system to recognize mathematical expressions by using tree transformation. In their system, they used a layout analysis technique for discovering baseline information, but, unfortunately, they gave neither a quantitative nor a qualitative analysis thereof.

The remainder of this paper is organized as follows: Section 2 describes the classification task. Section 3 introduces the features used in the classification task. Section 4 shows experimental results with very large databases. Finally, Section 5 derives a conclusion.

2 The Classification Task

Our task is the classification of each symbol into baseline and non-baseline class. In this task, several features were used. These features include (i) basic features based on the height, width, location, symbol type, category name, and entity name of the target symbol, and (ii) context features based on basic features of the symbols around the target symbol. A SVM was used to determine the class of each symbol.

The main contributions of this paper are two-fold: (i) showing high classification accuracy through a very large scale experiment, and (ii) showing the importance of the context features and symbol type for better classification.

3 Feature Extraction

3.1 Basic feature

For each symbol, we extract basic features to classify it into baseline and non-baseline. The basic features include width, height, location, category name, entity name and symbol type. Each symbol has a category name and an entity name. For example, symbol "γ" has category name "Greek" and entity name "gamma." The details about the category name and entity name can be found in [16]. The details for symbol types will be discussed below.

Note that we assume that the category of the symbol is given by some preceding symbol recognition method. This assumption is often acceptable because most mathematical OCRs are comprised of two steps as noted in Section 1.

3.1.1 Symbol types

A type is estimated for each symbol according to its occupation of three regions called the X, Y and Z regions. Figure 3 shows these regions. Region X is equivalent to the ascender part and region Z is equivalent to the descender part. Note that the specification of those regions are detailed in Section 3.1.2.

Estimation of type is very important to describe the variation in symbol sizes. For example, character "A" has type X and Y and character "a" has type Y. The estimation of types for mathematical symbols has no previous published literature, to the authors' best knowledge. All the past attempts have estimated only the types of alphanumeric characters. Characters are classified into 4 types according to ascender and descender parts.

To cover all the variation in symbol sizes, each of the X, Y and Z regions is divided into 4 subregions and, therefore, we have 12 regions. For example, in Fig 3, the symbol " $-$ " occupies 0.25 of the Y region, the symbol " $\in$ " occupies the full Y region, the symbol " $\cap$ " occupies the full Y region and 0.5 of the X region, and the symbol " $\int$ " occupies all of the X, Y and Z regions. In the proposed method, we have 24 symbol types as a different combination of 12 regions.

To estimate the type of symbol " $\cup$ " in Fig 4, the top and base of the Y region of the mathematical expression which included this symbol are first calculated using all the characters in this expressions; it is equal to the average of top and base of each individual character in the expression. For example, the base for character "A" is equal to its leftmost bounding box and its top can be calculated by subtracting this base value from the Y height. Then, the type is estimated according to these top and base values. Symbol " $\cup$ " in this expression has type "$X + Y + 0.5Z$."

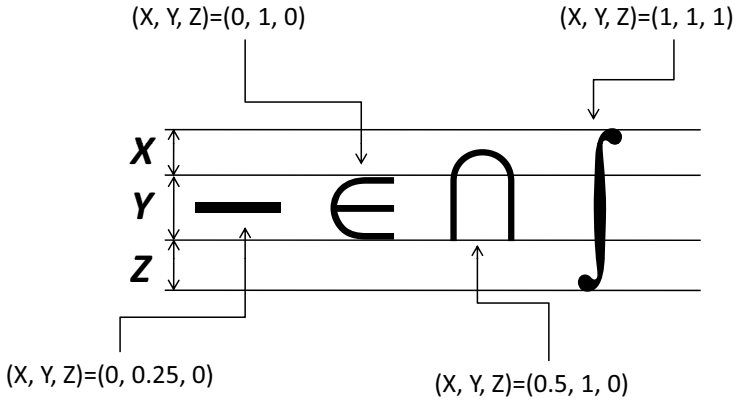


Figure 3: *X*, *Y*, and *Z* regions.

Figure 4: Example of the type for symbol "$\cup$."

3.1.2 Specifying *X*, *Y*, and *Z* regions

On setting the types for each symbol, we must know the height of the *X*, *Y*, and *Z* regions. In order to calculate the *X*, *Y* and *Z* heights of a document, the heights of the *X*, *Y* and *Z* for each baseline character of a document are first measured. At the measurement, we need to refer to its entity (i.e., recognition result). For example, if the entity name of a character is "A," the height of the *X* and *Y* regions are measured. Then the measured heights are averaged to calculate the *X*, *Y* and *Z* heights for all the baseline characters of the document. Note that the *X*, *Y* and *Z* heights are different in each document.

The *X*, *Y* and *Z* heights for non-baseline characters is also estimated in the same way. This is because they often have their own *X*, *Y* and *Z* heights (which is slightly different from the *X*, *Y* and *Z* heights of the baseline characters) due to their own font shapes [§].

3.2 Context features

Although basic features are important in the classification task, they are insufficient for some cases. For example, the symbols in Fig. 2 can not be classified directly using only basic features; the non-baseline symbols are closer in their height and position to baseline symbols. The features of the

[§]Readers may be confused by the fact that we need to discriminate between baseline characters and non-baseline characters for estimating their own *X*, *Y* and *Z* heights during the process toward our final goal, i.e., the classification task. For this discrimination we used a predetermined *X*, *Y* and *Z* heights which is calculated from all characters contained in the database. Of course, the result from this discrimination includes some errors. These errors do not affect the estimation seriously because we use the average of the heights.

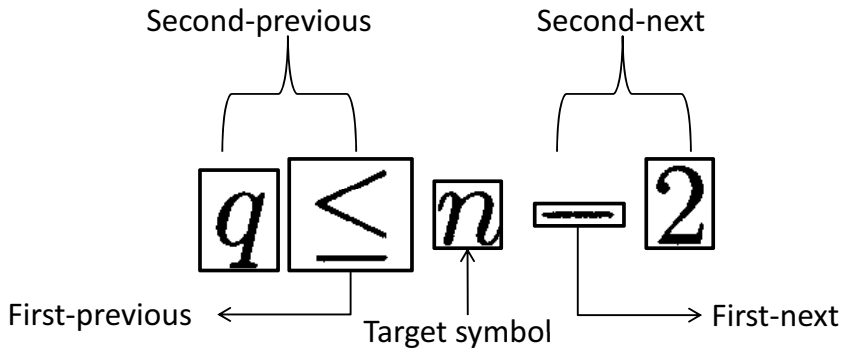


Figure 5: Example of context symbols for the target symbol "*n*."

symbols around the target symbol is very important in these cases. These features are called context features. Figure 5 shows the context symbols for symbol "*n*."

Context features are very important in the classification task; especially for mathematical symbols such as "−,""+," and " ∗ ." The basic features cannot classify these mathematical symbols exactly; their basic features overlap in baseline and non-baseline classes and, therefore, we used context features. These features include basic features for the symbols around the target symbol. As will be shown in the experimental results, these context features affect the classification accuracy.

4 Experimental Results

4.1 Database

The classification task was conducted on 229,898 target symbols. This huge number of symbols was extracted from two large databases, InftyCDB-1 [16, 17] and InftyCDB-2 [18], which together consist of 65 English articles (published between 1949 and 2000), 4 French articles (published between 1974 and 1988), and 7 German articles (published between 1956 and 1987) on pure mathematics. The total number of pages in the databases is 908.

To the authors' best knowledge, these databases are the largest of those used in past attempts on the classification task. For example, they are larger than the database used in [19], which consists of 297 pages. Such large databases are well suited to derive general properties of mathematical expressions and thus also suited to design the classifier for the baseline information.

All the mathematical expressions in the database were used, except matrices and fraction expressions. In these two types of expressions, the size of font is often very irregular and thus they will distort our results. Matrices and fraction expressions can be detected easily and treated separately from other mathematical expressions.

4.2 Classification accuracy

We used a binary SVM classifier to classify symbols into baseline and non-baseline symbol. To classify symbols, SVM determines a hyperplane which separate the two classes with maximum margin between the the vectors of the two classes. More details about SVM can be found in [20]. We use the LibSVM software [21] to train SVM classifiers. We construct SVM with linear kernel.

Table 1: Classification rate(%).

Category name	Without context features		With context features
	without type	with type	
Accent	93.39	90.80	96.23
Arrow	95.84	96.00	99.20
Big-symbols	99.93	99.93	99.70
Characters	97.80	99.29	99.42
Numeric	97.71	98.30	99.42
Operator	93.43	93.51	98.57
Others	89.66	91.56	96.92
Parentheses	96.69	99.72	99.60
Point	92.69	98.42	99.65
Average of accuracy	94.98	98.04	**99.25**

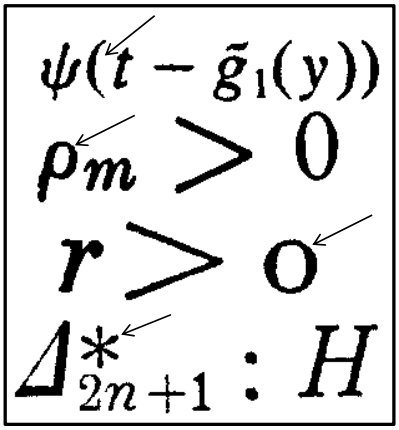


Figure 6: Example of misclassified symbols.

In the following table, we used 5-fold cross validation for the evaluation of the accuracy. The accuracy rate is evaluated for each category. We have 10 categories in the proposed method.

Table 1 shows the accuracy rate without using the context features. The accuracy rate is low in this case. From this table, we notice the effect of using symbol types in basic features as the accuracy rate is increased from 94.98 % to 98.04%.

Table 1 shows also the accuracy rate using the context features. The accuracy rate is increased in this case. From this table we notice that the effect of context features varies according to the category type. For example, the operator category is affected very much by the context features; its accuracy rate increased from 93.51 to 98.57 % when using the context features; its baseline classification depend on its preceding and succeeding symbols. In contrast, the parentheses category was not effected by the context features; its baseline classification does not depend on its preceding and succeeding symbols. This table shows the importance of using context features as the accuracy rate reached 99.25%. This accuracy rate was achieved with context of one symbol before and one after the target symbol.

Figure 6 shows an example of misclassified symbols in Table 1. A closer investigation of the misclassified symbols revealed that, (i) most of them occur in only two documents; these documents have a very special format. For example, the baseline parentheses in the first expression and the baseline symbol "ρ" in the second expression have very small height and they will be classified as non-baseline class. (ii) Some errors are due to the irregularity of symbol and characters. The details of irregular symbols and characters are found in [22]. For example, the baseline "0" in the third expression has very small height and it occupied Y region which differs from other files; normally, numeric symbols occupy X and Y regions. (iii) Some errors due to some symbols such as "$-$, " and "$*$; " these symbols have often difficult characteristics. For example, the non-base symbol " $*$ " in the last expression has large height and low position and will be classified as baseline symbol.

5 Conclusion

In this paper, each symbol is classified into baseline and non-baseline classes. To deal with the large variation of symbol sizes, the symbol type and the context features were used. A SVM classifier was used to classify each symbol into baseline and non-baseline class.

The performance of the classification task over very large databases is very high as it reached 99.25%. In this task, we emphasize the importance of both the symbol type and the context features to improve the performance.

Acknowledgment

The authors deeply thank the anonymous reviewers for their kind comments.

References and Notes

[1] K. Chan, and D. Yeung, "Mathematical expression recognition: A survey," Int. Journal Document Analysis and Recognition, vol. 3, pp. 3–15, 2000.

[2] R.H. Anderson, "Syntax-directed recognition of handprinted 2-D mathematics," pp. 436-459, Ph.D. Dissertation, Harvard University, Cambridge, 1968.

[3] M. Okamoto, and B. Miao, "Recognition of mathematical expressions by using the layout structure of symbols," Proc. 1st Int. Conf. Document Analysis and Recognition, pp. 242–250, 1991.

[4] H. Twaakyondo, and M. Okamoto, "Structure analysis and recognition of mathematical expressions," Proc. 3rd Int. Conf. Document Analysis and Recognition, pp. 430–437, 1995.

[5] H. J. Lee, and J. S.Wang, "Design of a mathematical expression system," Proc. 3rd Int. Conf. Document Analysis and Recognition, pp. 1084–1087, 1995.

[6] J. Ha, R. M. Haraalick, and I. T. Phillips, "Understanding mathematical expressions from document image," Proc. 3rd Int. Conf. Document Analysis and Recognition, pp. 956–959, 1995.

[7] H. J. Lee, and M.c. Lee, "Understanding mathematical expressions using procedure-oriented transformation," Int. Journal Pattern Recognition, vol. 24, pp. 447–457, 1994.

[8] A. Grbavec, and L. Pottier, "Mathematical formula recognition using graph grammar," Proc. Society of Photo-Optical Instrumentation Engineers, pp. 44–52, 1998.

[9] S. Lavirotte and L. Pottier, "Optical formula recognition," Proc. 4th Int.Conf. Document Analysis and Recognition, pp. 357–361, 1997.

[10] D. Blostein, and A. Grbavec, "Recognition of mathematical notation," In Handbook of Character Recognition and Document Image Analysis, World Scientific, pp. 557–582, 1997.

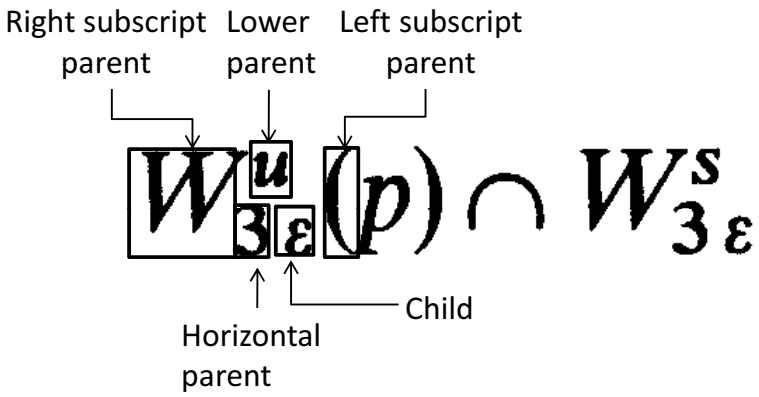


Figure 7: Example of the candidate parents for child "ε."

[11] U. Garain, and B. B. Chaudhuri, "A syntactic approach for processing mathematical expressions in printed documents," Proc. Int. Conf. Pattern Recognition, pp. 523–526, 2000.

[12] Y. Guo, L. Huang, C. Liu , and X. Jiang, "An automatic mathematical expression understanding system," Proc. 9th Int. Conf. Document Analysis and Recognition, pp. 719–723, 2007.

[13] R. Zanibbi, D. Blostein, and J.R. Cordy, "Baseline structure analysis of handwritten mathematics notation," Proc. 6th Int. Conf. Document Analysis and Recognition, pp. 768–773, 2001.

[14] J. -Y. Toumit, S. Garcia-Salicetti, and H. Emptoz, "A hierarchical and recursive model of mathematical expressions for automatic reading of mathematical documents," Proc. 5th Int. Conf. Document Analysis and Recognition, pp. 119–122, 1999.

[15] R. Zanibbi, D. Blostein, and J.R. Cordy, "Recognizing mathematical expressions using tree transformation," Int. Journal Pattern analysis and machine intelligence, vol. 24, pp. 1455–1467, 2002.

[16] M. Suzuki, S. Uchida, and A. Nomura, "A Ground-truthed mathematical character and symbol image database," Proc. 8th Int. Conf. Document Analysis and Recognition, pp. 675–679, 2005.

[17] S. Uchida, A. Nomura, and M. Suzuki, "Quantitative analysis of mathematical documents," Int. Journal Document Analysis and Recognition, vol. 7, pp. 211–218, 2005.

[18] M. Suzuki, C. Malon, and S. Uchida, "Databases of mathematical documents," Research Reports on Information Science and Electrical Engineering of Kyushu University, vol. 12, pp. 7–14, 2007.

[19] U. Garain and B. B. Chaudhuri, "A corpus for OCR research on mathematical expressions," Int. Journal Document Analysis and Recognition, vol. 7, pp. 241–259, 2005.

[20] V. N. Vapnik, "Statistical Learning Theory: A Primer," Int. Journal of Computer Vision, vol. 38, pp.9–13, 2000.

[21] C. C. Chang, C. J. Lin, LIBSVM: A library for support vector machines. ¡http://www.csie.ntu.edu.tw, 2001.

[22] A. Walaa , S. Uchida, M. Suzuki, "Automatic Classification of Spatial Relationships among Mathematical Symbols Using Geometric Features," Proc. 10th Int. Conf. Document Analysis and Recognition, Int. Journal the Institute of Electronics, Informations, and Communication Engineers, vol. E92-D, 2009

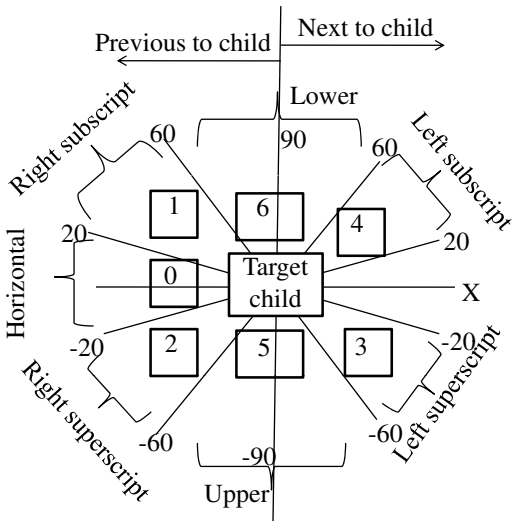


Figure 8: Angle between the child and its candidate parents.

Appendix

A Determine the parent of each child using SVM and baseline information

A.1 Outline

An SVM classifier also can be used with the baseline information to determine the parent of each child. For example, in Fig 7 symbol "ε" will have symbol "3" as parent from candidate parents. Selecting the parent for each symbol and the baseline information are important to specify the special relationship among symbols, the details about the special relationship can be found in [22].

The selection of the parent for each child from the candidate parents is done by the following steps. The symbols of mathematical expression are first ordered according to the leftmost bounding box. Then, the type of each symbols is estimated; the details about symbol types can be found in Section 3.1.1. After that, a normalized bounding box is evaluated for each symbol. The details about the normalized bounding box can be found in [22]. After that, the angle between the center of the normalized bounding box of the child and the center of the normalized bounding box for each symbol in the expression is calculated. Then, the first symbols, which satisfy the angle condition as illustrated in Fig 8, are selected as candidate parents. After that, the features are extracted from the child and its candidate parents. Finally, the SVM with quadratic kernel is used to select the parent for the child among 7 candidates (at most). Experiments using very large databases show high efficiency of the proposed method with the extraction accuracy reaching 98.54%.

Table 2: Classification rate(%) .

basic features only	98.47
with baseline	98.48
with angle	98.47
with H, D	98.49
with baseline and H, D	98.54

A.2 Feature extraction

We will divide features into basic features and additional features. Basic features include width, height for child and height for each of the candidate parent, and the horizontal distance between the child and each of the candidate parent. Additional features include baseline information for child and candidate parents, the angle between the child and the candidate parents, and relative size H and relative position D features. The details about H, D features can be found in [22].

A.3 Evaluation by cross-validation

An SVM classifier is used to select the parent of each child. In the experiment we have 187,415 children. We construct SVM with polynomial kernel with degree "2."

In Table 2, we used 5-fold cross validation for the evaluation of the accuracy. The accuracy rate is evaluated for each category. We have 10 categories in the proposed method. In these experiments we exclude all accent symbols because they have different characteristics which will confuse the classification task.

Table 2 shows the effect of using additional features in the proposed method. This table proves the importance of using baseline information and H, D features as the accuracy improved to 98.54%.

Audio/Visual/Tactual Presentation of Scientific Graphics

John A Gardner[1*], Vladimir Bulatov[1] Masakazu Suzuki[2], and Katsuhito Yamaguchi[3]

[1] ViewPlus Technologies, Inc,
Corvallis, OR 97333 USA
john.gardner@viewplus.com, vladimir.bulatov@viewplus.com

[2] Faculty of Mathematics, Kyushu University,
6-10-1 Hakozaki, Higashi-Ku, Fukuoka 812-8521, Japan suzuki@math.kyushu-u.ac.jp

[3] Nihon University, Junior College Funabashi Campus,
7-24-1, Narashinodai, Funabashi, Chiba 274-8501, Japan
eugene@gaea.jcn.nihon-u.ac.jp

Abstract

A review is given of the Audio/Tactile/Visual method for making graphical information universally accessible to all people including those who are blind or dyslexic or have other severe print disabilities. Current research is described for improving the Scalable Vector Graphic (SVG) format as well as SVG authoring and viewing software to make SVG amenable to excellent usability by all people.

1 Introduction

1.1 Brief Overview of information accessibility

Until a few decades ago, printed pages were synonymous with "information". Today information is understood in a much broader context. It is more and more likely to be delivered electronically even if still available in print. Electronic plain text is easily accessible to virtually every person who is capable of using a computer, and that is almost everybody. Screen reader software, good speech engines, and on-line braille displays make computers usable by people who are blind or have other print disabilities. Other software enhancements and hardware adaptations make them usable by people with virtually any physical disability. So "making information accessible" seems less and less necessary.

Unfortunately, plain text is not all there is to "information". Scientific information includes equations, and graphical information is ubiquitous. Math and graphics are not yet automatically accessible to computer users with print disabilities. Math presents a different accessibility challenge than does plain text. Math in electronic documents is often presented as an image that can be made automatically accessible only if there is an image ALT attribute. For example, Wikipedia (http://www.wikipedia.com) has the LaTeX equivalent of math expressions in the ALT attribute. This ALT attribute is spoken by screen readers, so Wikipedia math is more or less accessible. LaTeX source code is human-readable, but MathML source code is not. There have been studies of how to speak math both understandably and unambiguously [1] and a number of computer applications have been developed that present LaTeX [2,3] or MathML [4,5] in spoken, browsable audio. Some also

*Correspondence to:1853 SW Airport Avenue Corvallis, OR 97333, USA, TEL: +1 (541) 754-4002 x220, FAX: +1 (541) 738-6505.

can, at least in principle, present math equations in braille math codes on a braille display. Unfortunately most math is still published either as images without ALT attributes or in (inaccessible) PDF format.

Accessibility of graphical information presents an even more daunting challenge than math. Written descriptions of graphics have often been included in materials intended for people with print disabilities, but word descriptions are seldom as informative as the graphic. "A picture is worth a thousand words".

For some blind people, a tactile copy of a graphic can often provide as much information as the visual graphic does to sighted readers. Caveats are

- The graphic cannot be extremely complex since tactile access can never match vision for discerning small details.
- the blind person must generally be a good braille reader to read the text labels on the tactile graphic.
- the blind person needs to have learned how to interpret tactile graphic objects, a learning experience that is neither fast nor easy.
- the tactile graphic has been converted by translating text labels to braille and simplified when necessary to make the graphics amenable to tactile understanding, a process that can only partially be automated and requires a trained transcriber familiar with both tactile graphics techniques and the subject matter of the graphic.

Unfortunately, most blind people do not read braille, and many fewer have had any significant experience with tactile graphics. Since few blind people can understand tactile graphics, there is little incentive for transcribers to spend the time and money required to produce them. Consequently there are few examples for blind people to use to learn... - a classical chicken/egg problem.

1.2 Audio/visual/tactile graphics presentation technologies

About three decades ago, Prof. Donald Parkes [6,7] introduced a hybrid technique that combined audio and tactile information. A tactile graphic was mounted on a touch-sensitive pad connected to a computer. When a blind reader pressed on a point of interest, information was spoken by the computer. If the audio information was sufficient, most blind users could quickly learn to understand well-authored graphics. There have been no good studies of the subject, but conventional wisdom holds that elegant, well-organized graphics that are easy for sighted readers to grasp are usually also easy to read by audio/touch. Conversely, complex confusing graphics are confusing for all readers.Despite its obvious advantages, the audio/tactile method did not become broadly popular because the components were expensive, tactile graphics were difficult to make, and transcribers had a large learning curve to become competent to prepare the special computer information maps needed to correlate with "hot spot" positions on the graphic.

The Oregon State University Science Access Project (SAP http://dots.physics.orst.edu) took on the challenge of resolving these problems. The SAP was formed in 1990 to develop new information technologies that make possible universal information access. A major part of its mission was to ensure that mainstream graphical information could be created and distributed in a form that, like text and MathML math, is automatically accessible. The premise was that the electronic graphics file could be both the visual representation

and the audio computer map used for audio/tactile access and that tactile copy could be generated directly from the visual graphic without human intervention. Initial research [8] focused on the Virtual Reality Modeling Language (VRML), the only mainstream graphics language at the time even remotely capable of such use. When Scalable Vector Graphics (SVG) was developed by the World Wide Web community, the SAP immediately recognized its potential and adopted SVG as the language of universally usable graphics. It proved to be a good choice. This research topic eventually moved to the spin-off company ViewPlus Technologies, where accessible SVG was introduced in 2005 as the ViewPlus IVEO technology [9].

1.3 Refreshable Tactile Graphics Displays

Audio/touch access would be most convenient if a user had a good on-line refreshable tactile graphics monitor. Unfortunately there is still today no practical refreshable tactile graphics technology capable of representing general mainstream graphics. Pin matrix devices (PMD) [10] are technically feasible, although they are too expensive for any but specialty use. A PMD has pins that can be pushed up by solenoids or piezoelectric actuators, generally with pin resolution below 10 dpi (dots per inch), and always having only the binary possibility of pins pushed up or not pushed up. A PMD can sometimes usefully represent black and white line and block diagrams but not most modern color graphics. Presently there are two small commercially-available PMD's (http://www.kgs-jpn.co.jp) with resolution below 10 dpi and priced in the $10,000 price range. A German consortium [11] plans to introduce an A4 size PMD in the near future but has not announced the price.

Many research programs have been undertaken to develop better refreshable tactile technologies, and most have failed. A Chemistry research group at Pennsylvania State University may be succeeding. They have demonstrated prototype braille actuators [12] based on thin film piezoelectric polymers that could be used to make a modest-cost large scale tactile graphics display. The display is projected to have resolution better than 10 dpi, to have controllable dot height, and to be ready for commercialization within five years.

Dot height control is critical, because it enables height to be used as the tactile representation of intensity. Any graphic image that is visually understandable in gray scale is tactually understandable if dark regions are represented by tall dots and light regions by low dots. The only common graphical information for which a gray scale image is inadequate are such things as color-coded bar/column graphs and Geographic Information System (GIS) images. These color images can easily be made tactually distinguishable with a user option to represent colors as patterns.

1.4 Off-line Tactile graphics

Ten years ago it was possible to make tactile graphics on paper with "graphics mode" braille embossers or by using "capsule paper" [13]. Embossed tactile graphics could only be made using special proprietary software and typically had resolution approximately 10 dpi. These tactile graphics were useful only for low resolution line and block diagrams. Alternatively one could print or copy onto capsule paper and process it through an infrared heater causing black areas to rise. Higher resolution tactile graphics could be made this way, but the tactile images were still useful only for line and block graphics. Clearly, neither of these technologies was adequate for general transformation from mainstream to tactile form.

In 1996 SAP student Peter Langner invented a new high resolution embossing method [14]. ViewPlus Technologies was spun off to commercialize his invention along with other SAP-developed technologies. Langner's method was eventually expanded into the TIGER (Tactile Graphics EmbosseR) technology which was introduced commercially in 2000 and is a feature of all ViewPlus embossers. Tiger embossers have resolution of 20 dpi (which is beyond the static resolution of the average human finger), can emboss with eight dot heights, and have standard Windows printer drivers permitting them to emboss from any Windows application. By default, images are embossed in a tactile gray scale in which dark regions have big dots and light regions have small dots. There is a user option to replace colors by user-defined tactile patterns, and a user option for automatic generation of patterns will be introduced soon. Consequently nearly any mainstream image will emboss so that features are recognizable by touch. Of course additional audio information is needed for most of these tactile graphics to be understandable.

Figure 1 shows one author (JG) using IVEO Viewer to examine a periodic table. Figure 2 shows a typical GIS IVEO image of cancer statistics, and Figure 3 shows the dot pattern when that image is embossed on a Viewplus embosser.

2 Current state-of-the-art

2.1 The ViewPlus IVEO Viewer

The IVEO Viewer, which may be downloaded free from the ViewPlus web site, is a SVG viewer with special features making it appropriate for audio/braille/visual/tactile access to SVG graphics. A sighted user may mouse click to select text labels and hear them spoken. Selecting graphical objects causes their titles (contained in the object's SVG Title property) to be spoken. Double clicking causes the object's description field to be spoken. It is also possible to display the title of the full graphic and its description property in audio. The viewer permits normal SVG actions such as panning and zooming, all of which are accessible to a blind user. It has a search function for object titles and many other features that enhance access. A tactile copy may be obtained easily by printing to any ViewPlus embosser. A blind user would normally place this tactile copy on a touchpad or clamp it to a digital pen transceiver and then access the same information that sighted users obtain by clicking with a mouse. If the image is panned or zoomed, the new image may be embossed so the blind user can access the changed image. Graphics embossing typically requires 20 to 200 seconds depending on the embosser used and the size of the image. This "semi-on-line" access is less convenient than a refreshable tactile monitor but any blind person can use it without sighted assistance and get access in real time.

IVEO Viewer has a status bar whose font size is user-controllable. It permits people with reduced hearing and vision to optimize access by reading the text while it is being spoken. The status bar can also be displayed on an on-line braille display, making all audio information also accessible as braille. All spoken information may be browsed/reviewed by opening a review window. These features combine to provide maximum opportunity for anybody to understand the graphical information by using whatever combination of audio, braille, visual, and tactile modes is most useful.

Figure 1: Picture of John Gardner using IVEO Viewer to read the periodic table of the elements with a touchpad for input. The ViewPlus Emprint color embosser is also shown.

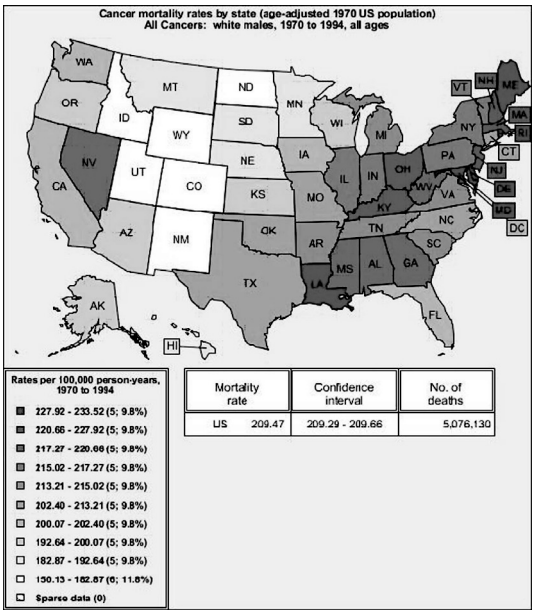


Figure 2: GIS display of total cancer mortality in the US. http://cancercontrolplanet.cancer.gov:8080/atlas/index.jsp

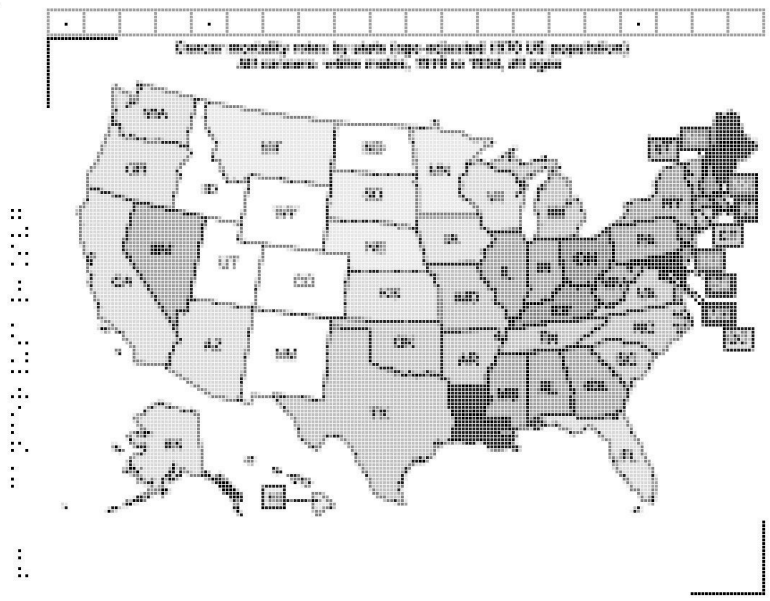


Figure 3: The dot pattern obtained when Figure 2 is printed to a ViewPlus® embosser. Dark dots represent big dots, light dots represent low dots.

2.2 Authoring accessible SVG Images

Adobe Illustrator, CorelDraw, MS Visio, and most other popular graphics authoring applications permit one to save-as-SVG. However the saved SVG file is minimally accessible to a blind person. The ViewPlus IVEO Creator and Creator Pro applications are needed to make SVG fully accessible. Either can open SVG files created by other applications. Creator Pro also permits one to import PDF, PostScript, bit map images, or to scan paper copy with a standard scanner. It also has a pseudo-printer feature permitting one to create SVG from any Windows application that can print. Creator Pro uses OCR to convert bit-mapped representations of text to proper SVG text.

The Creator applications permit users to improve text labels to make them optimally-accessible and to add titles and descriptions to the SVG file and to individual graphical objects. Not all semantically-meaningful portions of an SVG image are proper SVG objects. For example, a bit map imported into SVG is still a bit map and is the only SVG object in that image. Creator permits users to create invisible overlay objects so that any portion of the SVG image can be made selectable and given a title and description.

In principle SVG text labels should be automatically accessible, but in practice, they frequently are not. When one selects text, an entire span is selected and is spoken by IVEO Viewer. When a SVG file is authored in an IVEO Creator application or when non-SVG image is imported, most text is automatically gathered into semantically-appropriate spans and spoken properly. Most other authoring applications create SVG files in which semantically-meaningful phrases are either broken into several text spans or dumped into a span containing other text. Such text is often not understandable when selected and spoken in IVEO Viewer. Presently, human editing is required to repair it.

Scientific documents present even more serious difficulties in making text accessible. Greek characters, advanced math symbols, and other special scientific characters are generally put into separate spans by authoring applications. Sub- and superscripts are placed in individual spans. One can select and read these characters, but each Greek character, each superscript, and each subscript must be individually selected and read. Presently, human editors can improve accessibility of scientific text labels by pasting invisible overlays on equations and putting a word description of the equation into the title property.

3 Current Research on Improving SVG Accessibility

3.1 Improving accessibility of SVG text

Intelligent processing is required even for plain text in SVG authored with most applications if meaningful phrases are to be spoken when text is selected. It is possible in principle but difficult in practice to reconstruct plain text spans in an arbitrary SVG file. It is not even possible in principle to encode general math or scientific expressions with the very limited text structures presently permitted by SVG. Even if SVG was adequate, speech engines cannot speak such expressions. Both of these text problems can be eliminated elegantly by expanding the file format to permit plain text and general XML expressions to be associated with regions of the graphic. Thus it would be straightforward for an intelligent processor to create a text rendering for any plain text expression and associate it with the bounding box of that expression on the graphic. It is also possible to create a MathML rendering of a scientific expression and associate it with the position of that expression. An accessible viewer could then select and speak these expressions instead of whatever formal SVG text span is actually chosen.

The Infty group (http://www.inftyproject.org) and ViewPlus are working together to create the technologies needed to make these improvements. The Infty Reader math Optical Character Recognition) (OCR) application can recognize math expressions and transform them to either LaTeX or MathML with excellent accuracy. Their joint research is directed toward improving the IVEO Creator Pro application to add MathML to scientific SVG files. The improved software will also ensure that plain text labels on SVG graphics will be spoken properly.

IVEO Viewer will also be expanded by including the math-speaking algorithm from the ChattyInfty accessible math editor. ViewPlus will localize the routine to all languages in the worldwide ViewPlus market and consequently enhance the reach of IVEO and the ChattyInfty application.

3.2 Improving Accessibility of SVG Graphics

At present, humans must input information into the title and description properties of graphical objects. The need for graphics annotation is the single most difficult problem preventing graphics from being automatically fully accessible. Human input may forever be required for some types of graphical information, but there are several ways that annotations can be included automatically in more abstract information such as charts, diagrams, and GIS displays. For the near future, IVEO Creator capabilities enhanced by future Infty developments can increase accessibility of some types of graphical objects. The new Infty/IVEO Creator will be gradually improved to recognize graph axes, plain lines, dotted lines, dashed lines, open and filled data points of various shapes, etc.

More complete graphic object annotations can be included automatically by well-designed additions to charting/graphing authoring applications. Data set names, fitting curves notations, and other derived quantities can be given meaningful names by authors and be saved automatically in the title properties of objects representing these quantities. Both the accessibility and usefulness of electronic images can be greatly enhanced if the underlying data are also saved and correlated with position on the graphic. Such capabilities require further extension of SVG to add appropriate data namespaces. ViewPlus is presently making software to save in such extended SVG formats from ESRI GIS authoring software, Mathematica graph authoring applications, MS Excel charts, and its own Audio Graphing Calculator application. If all such graphical authoring applications eventually have the capability of saving in such formats, universally-accessible graphical information will no longer be a dream.

3.3 DAISY activities to extend SVG

DAISY (Digital Accessible Information SYstem http://daisy.org) is a consortium of nonprofit libraries and agencies serving needs of people who are blind, dyslexic, or who have other disabilities preventing them from reading normal print. DAISY has developed standards including an XML format for accessibility that is a superset of the popular EPUB electronic book specification, but it originally could not represent either math or graphics accessibly. One author (JG) was a member of a DAISY Working Group that developed a MathML extension that could make math accessible. It was formally adopted as part of DAISY XML in 2007. In that year the DAISY consortium formed an Accessible SVG Working Group and asked ViewPlus to join it to assist in developing authoring guidelines, to refine SVG by defining attributes promoting accessibility, and to extend SVG with additional DAISY namespaces when necessary. The Infty group has now become active in this arena and has joined the SVG Working group too. Most of these new specifications should be useful for many mainstream purposes, because they improve accessibility for everybody and additionally make images much more searchable and classifiable.

In addition to their collaboration in extending SVG, ViewPlus and Infty are both active in DAISY projects of their own. Infty is working to make their Infty software DAISY-compatible. ViewPlus has two DAISY related projects underway that are proving to be both a fertile source of ideas for DAISY SVG developments and an excellent testing ground for their usability. One is a ViewPlus collaboration with a group headed by the American Physical Society (APS)[15]. The goal of that collaboration is developing infrastructure so APS can publish its journals in DAISY format. The second is an ambitious project to develop universal DAISY curricula for kindergarten and elementary schools.[16]. Together, the two projects demonstrate use of DAISY from kindergarten to roket science.

Some of the new DAISY attributes are very specific to the needs of DAISY agencies. These specific attributes mostly focus on possibilities for creating tactile images independent of the visual image. For example a contour tactile image of a face could be created by an artist but cannot presently be reliably computer-derived from the visual image.

More general DAISY attributes include several that permit objects to be grouped into categories such as countries, states/provinces, large cities, capital cities, etc. on a North America map. DAISY levels are independent of SVG object parent/child structure. This is a major advantage since one can impose DAISY structure easily on essentially any SVG image. Restructuring the SVG itself is often very difficult.

DAISY is also adding additional namespaces to extend the information beyond what can

be included in standard SVG. Standard SVG has a single title and description property for the file and for individual graphical objects, and these may contain plain text only. This is a severe restriction for scientific graphics. For example it would be desirable for the SVG file description property to contain the figure caption. Presently this is not possible, because scientific figures frequently have math and other nonstandard text characters in the caption. With DAISY additions, the file, individual graphical objects, and text fields all now have an unlimited number of description fields capable of containing XML information. XML information may also be associated with arbitrary regions of the graphic, and quantitative data may be included and associated with regions of the graphic. These new namespaces are necessary for the information that will be included by the ViewPlus/Infty collaborative development and by the "save-as DAISY SVG additions under development by ViewPlus. Many of these new features will be incorporated into either the American Physical Society DAISY SVG images or the ViewPlus DAISY curricula.

None of these expansions affects the display of SVG in current mainstream SVG viewers, but they permit special ones like IVEO Viewer to provide much improved accessibility. It is likely that mainstream users will eventually adopt a specialty viewer or that mainstream viewers will expose at least some of the additional DAISY information. When APS journals begin to include the actual data in a significant number of the images, scientists will certainly use software that permit them to extract these data and use it in different models, to compare with similar experiments, etc. This is the most obvious new capability that make accessible figures of immediate interest to mainstream readers.

Acknowledgements

ViewPlus SVG developments are supported in part by Small Business Innovation Research (SBIR) grants from the National Eye Institute of the US National Institutes of Health.

References and Notes

[1] Karshmer, A.I., and Gillan, D., How well can we Read Equations to Blind Mathematics Students: Some Answers from Psychology, Proceedings of the 2003 Human Computer Interface International Conference, Crete, Greece, July, 2003.

[2] TV Raman, Audio System for Technical Recordings (ASTeR), PhD Thesis, Cornell University, 1994, http://www.cs.cornell.edu/home/raman/aster/demo.html

[3] Katsuhito Yamaguchi and Masakazu Suzuki, Math-Document Accessibility with Infty-Reader and Chatty-Infty, Proceedings of the 2008 CSUN Conference on Technology and Persons with Disabilities, Los Angeles, March, 2009, http://www.letsgoexpo.com/Utilities/File/viewfile.cfm?LCID=1462&eID=80000093

[4] Soiffer, N., Accessible Mathematics Made Easy, Proceedings of the 2004 CSUN conference on Technology and Persons with Disabilities, Los Angeles, March, 2004. November, 2009 at www.csun.edu/cod/conf/2004/proceedings/177.htm

[5] Karshmer, A., Bledsoe, C., and Stanley, P., The Architecture of a Comprehensive Equation Browser for the Print Impaired, Proceedings of the IEEE Symposium on Visual Languages and Human Centric Computing, Dallas Texas, September, 2005

[6] Parkes D 1988, "Nomad: an Audio-Tactile Tool for the Acquisition, Use and Management of Spatially Distributed Information by Partially Sighted and Blind Persons", eds Tatham AF and Dodds AG, Proceedings of the Second International Symposium on Maps and Graphics for Visually Handicapped People, King's College, University of London, pp. 24-29

[7] Parkes 1991, "Nomad: Enabling Access to Graphics and Text Based Information for Blind, Visually Impaired and Other Disability Groups", Conference Proceedings, Vol. 5. World congress on Technology 1991, Arlington, Virginia, pp. 690-714

[8] Non-Visual Access to Non-Textual Information through DotsPlus and Accessible VRML" John A. Gardner and Vladimir L. Bulatov Proceedings of the 15th IFIP World Computer Congress, Vienna, September 1998

[9] The ViewPlus IVEO Technology for Universally Usable Graphical Information, John A. Gardner, Vladimir Bulatov, and Holly Stowell, Proceedings of the 2005 CSUN International Conference on Technology and People with Disabilities, Los Angeles, CA, 16-19 March, 2005 http://www.viewplus.com/about/abstracts/05csungardner2.html

[10] Vidal-Verdu, F. and Hafez, M. (2007) Graphic Tactile Displays for Visually-Impaired People. IEEE Transactions in Neural Systems and Rehabilitation Engineering, 15(4): 119-130.

[11] Völkel, T.; Weber, G.; Baumann, U. (2008) Tactile Graphics Revised: The Novel BrailleDis 9000 Pin-Matrix Device with Multitouch Input. In: K. Miesenberger et al. (eds.): Proceedings of the International Conference on Computers Helping People With Special Needs (ICCHP'08), LNCS 5105, pp. 835-842, : http://dx.doi.org/10.1007/978-3-540-70540-6_124

[12] K. Ren, S. Liu, M. Lin, Y. Wang, Q. Zhang, A compact electroactive polymer actuator suitable for refreshable Braille display Sensors and Actuators A 143 (2008) 335–342 (2007)
www.sciencedirect.com

[13] Gardner 1996, "Tactile Graphics, an Overview and Resource Guideh, John A. Gardner, published in Information Technology and Disabilities, an electronic journal, http://www.rit.edu/ easi/itd/itdv03n4/article2.html. An updated well-hyperlinked version is also available at http://dots.physics.orst.edu/tactile/tactile.html

[14] "Tiger, A New Age Of Tactile Text And Graphics" Patricia Walsh and John A. Gardner Proceedings of the 2001 CSUN International Conference on Technology and Persons with Disabilities, Los Angeles, CA, March 21-24, 2001

[15] Making journals accessible to the visually impaired: the future is near. John A. Gardner, Vladimir Bulatov, and Robert A. Kelly. Learned Publishing 22(4) 2009 pp.314-319. http://www.viewplus.com/about/abstracts/09learnpubgardner.html

[16] ViewPlus Math and Science Curricula with Fully Accessible Illustrations. John Gardner, CarolynGardner, Blake Jones,, Kayleen Hagen, and Tara Callaway. Proceedings of the 2009 International Conference on Technology and Persons with Disabilities, Los Angeles, CA, 18-21 March, 2009. http://www.viewplus.com/about/abstracts/09csungardner2.html

Orientation-Independent Recognition of Handwritten Characters with Integral Invariants

Oleg Golubitsky, Vadim Mazalov and Stephen M. Watt

The University of Western Ontario
London, Ontario, Canada N6A 5B7
ogolubit@uwo.ca, vmazalov@uwo.ca, watt@uwo.ca

Abstract

We present an approach to recognize handwritten characters independently of their orientation. The method is based on the theory of integral invariants and yields good results in classifying rotated samples. We propose two recognition techniques taking advantage of integral invariants up to second order. Truncated Legendre-Sobolev series are used to represent the invariant functions and recognition is based on proximity to the local convex hulls of known classes. We compare performance of these new methods with another widely used recognition method based on geometric moment invariants. The results obtained indicate that integral invariants give better recognition rates with less computation, confirming they are suitable for classification of rotated handwritten characters in a pen-based environment.

1 Introduction

We are interested in robust methods for the recognition of handwritten mathematical symbols. We view the trace of the symbol, as it is written, as a two-dimensional curve made up of a number of continuous segments and treat recognition as a classification problem. This subject of classifying two-dimensional parametric curves has been gaining importance in recent years. Moreover, mathematical handwriting recognition has received increasing attention with the popularity of hand-held mobile and digital tablet devices [1]. In this setting the accuracy and speed of an online character classification algorithm is important.

There are a number of factors that give the recognition of mathematics additional challenges beyond that of normal text recognition. Among these, we can observe the relatively large "alphabet" of similar looking few-stroke symbols. It is normally the case that symbols tend to be well isolated. There is no fixed dictionary of multi-symbol "words," but it is possible to identify expressions that occur more often in particular fields [2]. In addition, mathematical expressions are two-dimensional objects and the placement of symbols is important in contextual analysis [3]. Character classification algorithms for handwritten mathematics recognition therefore need special consideration.

It has been shown earlier how to classify a curve represented by truncated expansions of its coordinate functions in orthogonal bases [4, 5, 6, 7, 8]. Different bases have been considered, including Chebyshev, Legendre and Legendre-Sobolev bases. Most recently, attention of this research program has focused on Legendre-Sobolev series. These are easy to compute and provide an useful distance measure in the first jet space, taking derivatives into account. Test results confirm that this technique is indeed effective and allows to achieve 97.5% recognition rate.

We now address the problem that the recognition rate may be undermined by the variation in orientation of individual symbols. This may be more of a problem when symbols are written in a well-separated manner than when text is written cursively. Orientation variation is usually addressed by "de-slanting" symbols with a transformation on the coordinate space. One difficulty with this approach in a mathematical handwriting setting is

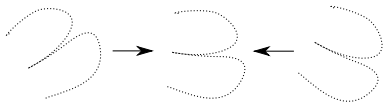


Figure 1: Rotation of a symbol

that different characters may require different correction and the degree of correction is not known in advance. With mathematical handwriting, it can be difficult to detect dominant orientation from symbol features.

Different solutions have been proposed, usually dealing with *ad hoc* rotation of a character after it is completely written (Figure 1). This rotation, as well as symbol resizing, are performed during a normalization stage in most of the online techniques. We propose a different approach: rather than rotating a sample by some estimated amount, we compute from the sample certain functions that are invariant under rotation. We ask to what extent these transformations affect the classification rate and present new algorithms for classifying symbols in the presence of such transformations. We consider methods using classification with integral invariants (CII) and classification with coordinate functions and integral invariants (CCFII). For these we use the theory of integral invariants of parametric curves [9]. To objectively evaluate recognition rate of the proposed techniques, we compare to a similar algorithm that uses geometric moment functions for the rotation-independent classification. We call this last method classification with coordinate functions and moment invariants (CCFMI).

In our methods, curves are represented as points in a vector space formed by the coefficients of their approximating truncated Legendre-Sobolev series. For CII, we take the integral invariants as the curves to be approximated and look for nearest classes in a manner we describe below. For CCFII, the top N classes are selected with integral invariants, then the sample is rotated to determine the angle which gives minimal distance based on coordinate curves. The CCFMI method is similar to CCFII except that it computes geometric moment invariants to obtain top N candidates. The proposed algorithms are online in the sense that most of the computation is performed while the sample is written, with minor overhead after pen-up. The algorithms are as well independent of translation and scaling, which is achieved by dropping the constant terms from the series and by normalizing the coefficient vectors, respectively.

This paper is organized as follows. In Section 2 we summarize the theory of integral invariants as applied in our algorithms. In Section 3 we outline concepts of geometric moments and give examples of moment invariants. Algorithms based on integral invariants are described in detail in Section 4. In Section 5 we present the CCFMI algorithm, relying on explanations in Section 4. Experimental settings and performance comparison are given in Section 6. In the conclusion we discuss causes of misclassification and outline directions for improvement of the proposed methods.

2 Integral Invariants

Integral invariants provide an elegant approach to planar and spatial curve classification under affine transformations. In terms of handwriting recognition, a symbol is given as a parameterized piecewise continuous curve defined by a discrete sequence of points. For a symbol we compute certain integral quantities from the coordinate functions, which are

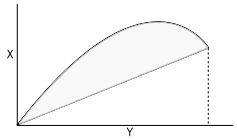


Figure 2: Geometric representation of the integral invariant of the first order

Figure 3: Ambiguity, introduced by shear and rotation

then also functions of the curve parameterization. Exposing the sample to transformations results in the same invariant functions. As opposed to differential invariants, such integral invariants are relatively insensitive to small perturbations, and are therefore applicable to classification of handwritten characters with sampling noise.

As the name suggests, integral invariant functions are constructed from quantities obtained by integration. We consider the following invariants, defined in terms of the coordinate functions $X(\lambda)$ and $Y(\lambda)$:

$$I_0(\lambda) = \sqrt{X^2(\lambda) + Y^2(\lambda)} = R(\lambda),$$

$$I_1(\lambda) = \int_0^\lambda X(\tau)dY(\tau) - \frac{1}{2}X(\lambda)Y(\lambda)$$

$$I_2(\lambda) = X(\lambda)\int_0^\lambda X(\tau)Y(\tau)dY(\tau) - \frac{1}{2}Y(\lambda)\int_0^\lambda X^2(\tau)dY(\tau) - \frac{1}{6}X^2(\lambda)Y^2(\lambda).$$

In our case $X(\lambda)$, $Y(\lambda)$ are parameterized by Euclidean arc length. The invariant $I_1(\lambda)$ can be interpreted geometrically as the area between the curve and its secant (Figure 2). The derivation of these is developed in [9].

Functions $I_1(\lambda)$ and $I_2(\lambda)$ are invariant under $SL(2)$, the group of special linear transformations, while $I_0(\lambda)$ is invariant under the action of the special orthogonal group $SO(2)$. An invariant under the full affine group can be constructed as the quotient $I_2(\lambda)/I_1^2(\lambda)$. This is, however, less stable to compute than I_2. Instead, by translating the origin and normalizing the size of the sample we can restrict our attention to the $SL(2)$ invariants without loss of generality.

Among the actions of special linear group, two are of particular interest in handwriting recognition: rotation and shear transformations. The latter, in its full generality, is a separate nontrivial problem and is not considered in this paper. One of the difficulties is in choosing the appropriate shear-invariant parameterization of the coordinate functions. Another problem, arising with mathematical alphabets, is the requirement of careful analysis of transformation limits to avoid blending classes. For example, a sample L (which initially can be confused with $\lfloor$), when exposed to shear transformation, becomes a subject to misclassification with $\angle$. If, in addition, we allow arbitrary rotation and scaling of the character, the set of matching candidates will include $<$, $>$, 7, V, $\bigwedge$, $\rceil$, $\surd$, Γ and $\wedge$ (Figure 3). Therefore, our attention is currently drawn to the action of the subgroup $SO(2)$. We note that in practice shear is seen most on tall, thin symbols and this can to an extent be corrected by rotation.

The invariant representation of a symbol curve is approximated with Legendre-Sobolev polynomials. Coefficients of the truncated Legendre-Sobolev expansion are used to construct a point for each character in a training set annotated with ground truth. Classification is performed based on the distance from the test point to the convex hull of points in the nearest classes.

Our experiments show that using I_2 gives only a minor increase in the recognition rate (about 1%) in CII, but increases the computational cost significantly. As for CCFII, the nature of the algorithm makes the accuracy of the integral invariant classification less critical. Invariant functions are used only for the purpose of selecting top N candidate classes, which are subsequently analyzed. The function obtained with invariants I_0 and I_1 was therefore chosen as sufficient.

3 Geometric Moments

Similar to integral invariants, moment invariants provide a framework to describe curves independently of orientation. Among moment functions one can select geometric, Zernike, radial and Legendre moments [10]. For the purpose of online curve classification under pressure of computational constraints, geometric moments are of special interest since they are easy to calculate, while invariant under scaling, translation and rotation.

Having been introduced by Hu [11], geometric moments are widely used for shape and pattern classification [10, 12, 13]. A $(p+q)$-th order moment of f can be expressed as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y)$$

In general, translation invariance is achieved by computing central moments

$$\mu_{pq} = \sum_x \sum_y (x-x_0)^p (y-y_0)^q f(x,y),\ x_0 = \frac{m_{10}}{m_{00}} \text{ and } y_0 = \frac{m_{01}}{m_{00}}$$

and scale normalization is performed as

$$\eta_{pq} = \mu_{pq} / (\mu_{00})^{(p+q+2)/2}$$

The first three moment invariants are derived from algebraic invariants and can be represented as

$$M_1 = \eta_{20} + \eta_{02}, \quad M_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2, \quad M_3 = \eta_{20}\eta_{02} - \eta_{11}^2.$$

Independence of orientation of the above expressions can be verified by substitution with the geometric moments obtained after rotation transformation

$$\begin{aligned}
m'_{20} &= \frac{1+\cos 2\alpha}{2} m_{20} - \sin 2\alpha\ m_{11} + \frac{1-\cos 2\alpha}{2} m_{02}, \\
m'_{11} &= \frac{\sin 2\alpha}{2}\ m_{20} + \cos 2\alpha\ m_{11} - \frac{\sin 2\alpha}{2}\ m_{02}, \\
m'_{02} &= \frac{1-\cos 2\alpha}{2} m_{20} + \sin 2\alpha\ m_{11} + \frac{1+\cos 2\alpha}{2} m_{02}.
\end{aligned}$$

One can omit translation and scale normalization of moments by normalizing a sample's coordinates first. In this case the moment invariants are derived in terms of moments m_{pq}.

4 CII and CCFII

Consider the coordinate functions $X(\lambda)$ and $Y(\lambda)$ of a single- or multi-stroke sample. Multi-stroke symbols are represented by the coordinate functions of consecutively joined strokes. The first step to approximate $X(\lambda)$ and $Y(\lambda)$ as truncated series in basis of Legendre-Sobolev polynomials. These polynomials are orthogonal with respect to Legendre-Sobolev inner product

$$\langle f, g\rangle = \int_a^b f(\lambda)g(\lambda)d\lambda + \mu\int_a^b f'(\lambda)g'(\lambda)d\lambda$$

where the functions $f(\lambda)$ and $g(\lambda)$ are differentiable on the interval $[a, b]$, μ is a numeric parameter and can be chosen experimentally. It has been shown in [14] that $\mu = 1/8$ yields good classification results.

Let x_0, x_1,...,x_d be the coefficients of the approximation for $X(\lambda)$ and similarly for $Y(\lambda)$. Note, that these coefficients are computed while the curve is written with a small constant time overhead after pen-up [7]. We take $d = 12$, because it allows us to achieve accurate enough approximation with error unnoticeable to a human [8].

Since the first polynomial (for any inner product) is 1, point (x_0, y_0) can be thought of as the curve's center. We can therefore normalize the curve with respect to position by simply discarding the first coefficients. Scale normalization is performed by normalizing the vector $(x_1, ..., x_d, y_1, ..., y_d)$, taking advantage of the fact that the norm of the vector is proportional to the size of the curve, to obtain $(\bar{x}_1, ..., \bar{x}_d, \bar{y}_1, ..., \bar{y}_d)$.

With this approximation, the integral invariant functions take the form

$$I_0(\lambda) = \sqrt{\left(\sum_{i=1}^{d}\bar{x}_i P_i(\lambda)\right)^2 + \left(\sum_{i=1}^{d}\bar{y}_i P_i(\lambda)\right)^2},$$

$$I_1(\lambda) = \sum_{i,j=1}^{d}\bar{x}_i\bar{y}_j\left[\int_0^\lambda P_i(\tau)P_j'(\tau)d\tau - \frac{1}{2}P_i(\lambda)P_j(\lambda)\right]$$

Here P_i denotes the i-th Legendre-Sobolev polynomial.

A similar process of approximation is then applied to the invariant functions, yielding a 24-dimensional vector for each sample $(\bar{I}_{0,1}, ..., \bar{I}_{0,d}, \bar{I}_{1,1}, ..., \bar{I}_{1,d})$. Taking the second term in the expression for $I_1(\lambda)$ as precomputed, the Legendre-Sobolev coefficients can be calculated quickly, in time quadratic in d. The coefficients for $I_0(\lambda)$ are computed in the same way.

Classification is based on evaluation of the distance from the sample to the convex hulls of the nearest neighbours and selecting the classes with the smallest distance. Different distance measures were considered in previous work [8] with emphasis on fast computation. Manhattan distance was chosen as the most efficient for pre-classification (selecting the nearest neighbours), while square Euclidean distance gave a lower error rate when used as the distance from a sample to the convex hulls.

Computing the distance from a point to a convex hull can be expensive. We were able to specialize the problem by taking the convex hull to be a simplex, since the number of nearest neighbours in our algorithms is less than dimension of the vector space and the points are in generic position. If the points are not in generic position, a minor perturbation is performed, only slightly affecting the distance. We then apply the algorithm recursively to find the projection from the point on the smallest affine subspace containing the simplex, until the projection happens to be inside the simplex. On each iteration the projection is expressed as a linear combination of the vertices of the simplex, considering the vertices

with non-negative corresponding coefficients. The complexity of this algorithm is $O(N^4)$, where N is the dimension. In practice it performs much faster, since at each recursive call the dimension often drops by more than one [14].

The CII algorithm relies on approximation of the invariant functions, as described above. We select the class closest to the sample in the space of coefficients of truncated polynomial series. The algorithm does not depend on the number of classes, since only one class is considered.

As an alternative, in CCFII the coefficients $(\bar{I}_{0,0}, \ldots, \bar{I}_{0,d}, \bar{I}_{1,0}, \ldots, \bar{I}_{1,d})$ are used to select the closest N candidate classes. The value for N may be determined empirically to ensure high probability of the correct class being within the ones chosen. Having a fixed small number of classes with the correct class among them, we evaluate the minimal distance from the sample to each class with respect to various sample rotations. This procedure gives correct class as well as the rotation angle. The angle is determined as the solution to the minimization problem

$$\min_{\alpha} \left(\sum_k (X_k - (x_k \cos\alpha + y_k \sin\alpha))^2 + \sum_k (Y_k - (-x_k \sin\alpha + y_k \cos\alpha))^2 \right),$$

where X_k, Y_k are the coefficients of the Legendre-Sobolev approximation of the coordinate functions of the training symbols, and x_k, y_k are the coefficients of the test sample. The global minimum is selected among the output of the function at the boundary points of the closed interval α and at the stationary point

$$\alpha = \arctan\left(\frac{\sum_k (X_k y_k - Y_k x_k)}{\sum_k (X_k x_k + Y_k y_k)} \right).$$

5 CCFMI

The $(p+q)$-th moment functions of a sample's coordinates can be expressed as

$$m_{pq}(\lambda_\ell) = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} X(\lambda_i)^p Y(\lambda_j)^q f\big(X(\lambda_i), Y(\lambda_j)\big)$$

where $X(\lambda_i)$ and $Y(\lambda_i)$ are the coordinates X and Y at sample point i. We have taken the intensity function to be of the form $f\big(X(\lambda_i), Y(\lambda_j)\big) = \sqrt{X(\lambda_i)^2 + Y(\lambda_j)^2}$ and work directly with moments, since normalization with respect to size and position is already performed in the algorithm. Specifically, we tested the following rotation invariants

$$\begin{aligned} M_0(\lambda) &= m_{00}(\lambda), \\ M_1(\lambda) &= m_{20}(\lambda) + m_{02}(\lambda), \\ M_2(\lambda) &= (m_{20}(\lambda) - m_{02}(\lambda))^2 + 4m_{11}(\lambda)^2. \end{aligned}$$

As in CCFII, CCFMI selects the top N classes with rotation invariant functions. To make a fair comparison, we considered the classification rate for two combinations of moment invariants: $M_0(\lambda)$, $M_1(\lambda)$ and $M_1(\lambda)$, $M_2(\lambda)$. Classification with $M_1(\lambda)$, $M_2(\lambda)$ in general gave 3% higher error rate. We therefore focused our attention on improving the recognition rate of $M_0(\lambda)$ and $M_1(\lambda)$ by variation of number of classes and number of nearest neighbours. Details of our experiments are described in the following section.

Table 1: Presence (%) of the correct class within the top N classes, CCFII

$N=1$	2	3	4	5	6	7	10	15	**20**	25
87.9	95.1	96.8	97.7	98.3	98.7	98.9	99.4	99.5	**99.5**	99.5

Table 2: Error rate (%), depending on number of nearest neighbours, CCFII

angle (radians)	$K=8$	10	12	14	16	18	19	**20**	21	22
0	4.4	3.9	4.2	4.0	3.9	3.9	3.8	**3.7**	3.8	3.8
0.3	6.2	5.7	5.7	5.4	5.4	5.4	5.3	**5.3**	5.4	5.4
0.5	7.4	6.9	6.8	6.7	6.6	6.5	6.4	**6.4**	6.5	6.5
0.7	8.5	7.9	7.7	7.6	7.4	7.4	7.2	**7.2**	7.3	7.4
0.9	9.3	8.8	8.6	8.3	8.2	8.2	8.2	**8.1**	8.2	8.2
1.1	9.6	9.0	8.7	8.6	8.4	8.4	8.2	**8.2**	8.4	8.4
average	7.5	7.0	7.0	6.8	6.6	6.6	6.5	**6.5**	6.6	6.6

6 Experimental Details and Evaluation of Results

Our dataset comprised 50,703 handwritten mathematical symbols from 242 classes. All samples were represented in a uniform InkML format [15] and stored in a single file. Each symbol definition included the number of strokes and the (X, Y) coordinates of the trace sample points. For some symbols we also had information about timing, pen-up strokes, pen pressure and context (the formula containing the symbol). This additional information was not used in the present experiment.

All symbols had been inspected visually in order to discard samples unrecognizable by a human. Symbols that looked to a human reader as belonging to more than one class were labeled with all those classes. Classes that were indistinguishable without context were merged. For example, we united the classes "Capital O, little o, omicron, zero", capital Greek letters with Latin analogues, etc. As a result, 38,493 samples had one class label, 10,224 samples had 2 class labels, 1,954 samples had 3, 19 samples had 4, and 13 samples had 5. This resulted in 378 composite sets.

To implement 10-fold cross-validation we randomly divided the dataset into 10 parts, preserving the proportions of class sizes. The normalized Legendre-Sobolev coefficient vectors of coordinate functions of randomly rotated symbols, as well as coefficients of integral invariants were pre-computed for all symbols. See [8] for more detailed description of the experimental settings.

According to our tests, CII gives a 88% recognition rate. This recognition rate does not depend on the angle to which test samples are rotated. Neither does the frequency of occurrence of the correct class in the top N classes depend on rotation angle.

To measure the best performance of CCFII, we first determined experimentally the number N of top classes required to contain the correct class most of the time. The results obtained are shown in the Table 1. We find $N = 20$ to be an appropriate balance between accuracy and the complexity introduced by integral invariants. With fixed N, the relationship between the number of nearest neighbours K and the error rate for different

Table 3: Presence (%) of the correct class within the top N classes, CCFMI

$N=1$	2	3	4	5	10	20	30	40	**50**	55
51.5	68.3	77.2	82.2	85.9	95.3	98.8	98.9	99.0	**99.0**	99.0

Table 4: Error rate (%), depending on number of nearest neighbours, CCFMI

angle (radians)	$K=8$	10	12	14	16	18	20	21	**22**	23
0	7.0	6.6	6.4	6.2	6.1	6.1	5.9	5.8	**5.8**	6.0
0.3	8.0	7.8	7.6	7.4	7.2	7.1	7.0	7.2	**7.1**	7.2
0.5	9.3	9.1	8.9	8.5	8.3	8.3	8.2	8.2	**8.1**	8.3
0.7	10.4	10.1	9.9	9.5	9.4	9.2	9.1	9.2	**9.2**	9.3
0.9	11.5	11.1	10.8	10.4	10.2	10.2	10.1	10.0	**10.0**	10.0
1.1	11.4	11.1	10.7	10.4	10.2	10.1	10.1	10.0	**10.0**	10.0
average	9.6	9.3	9.1	8.7	8.6	8.5	8.4	8.4	**8.4**	8.5

Table 5: Error rates of CII, CCFII and CCFMI

α, rad.	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	1.0	1.1
CII	12	12	12	12	12	12	12	12	12	12
CCFII	**3.7**	**3.9**	**4.5**	**5.3**	**5.9**	**6.4**	**6.6**	**7.2**	**8.2**	**8.2**
CCFMI	5.8	5.9	6.5	7.1	7.7	8.1	8.7	9.2	10	10

angles is shown in the Table 2. We observed that CCFII gives the best recognition rate for $K = 20$. In this framework, CCFII's rate starts at 96.3% for non-rotated samples and decreases slightly with increase in angle, but never approaches CII (see Table 5).

A similar approach was taken to measure the performance of CCFMI. We first measured the number of classes ($N = 50$) required to contain the correct class most of the time (Table 3) and then found the K that yields the best classification results (Table 4).

A comparison of the performance of CCFII and CCFMI is presented in Figures 4 and 5. Relative classification results are shown in Table 5 and Figure 6. We see that CCFII has a better error rate, while requiring fewer candidate classes and fewer nearest neighbour computations.

7 Conclusion

We have presented methods to classify handwritten characters, independently of orientation, based on integral invariants and have compared them with classification using geometric moment invariants. We have observed that integral invariants perform better while requiring less computation. We therefore conclude that integral invariants are a suitable instrument in the recognition of handwritten characters when orientation is uncertain.

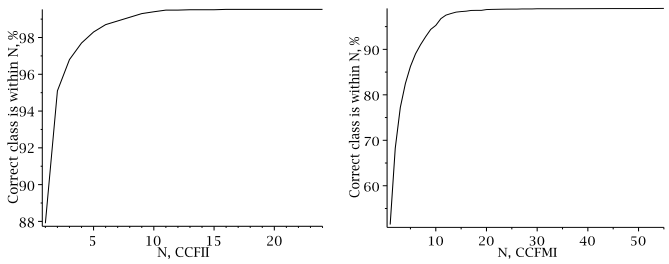


Figure 4: Presence of the correct class within N for CCFII (left) and CCFMI (right)

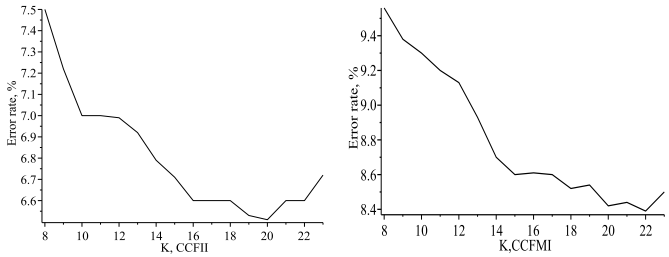


Figure 5: Error rate for different K for CCFII (left) and CCFMI (right)

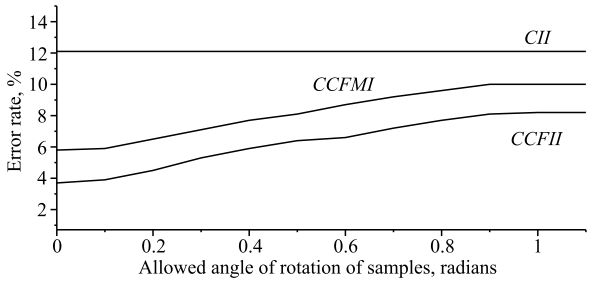


Figure 6: Error rates of CII, CCFII, CCFMI

As expected, we noticed an increase in error rate with the rotation angle for CCFII and CCFMI (Figure 6). The typical misclassifications that arise are when distinct symbols have similar shape and are normally distinguished by their orientation, for example "1" and "/", "+" and "×", "U" and "⊂". As a possible solution to this, a system could consider the tendency to write characters in similar orientations and restrict the range of angles for nearby symbols. A technique similar to CCFII can be applied to classify symbols as part of an expression with small adjustments to the minimization function. This approach has a number of other benefits, such as contextual and notational analysis, and should be considered as a logical continuation of the work presented.

References and Notes

[1] E. Smirnova and S.M. Watt, Communicating Mathematics via Pen-Based Computer Interfaces, In *Proc. 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, (SYNASC 2008)*, IEEE Computer Society, pp. 9-18, 2008.

[2] S.M. Watt, An Empirical Measure on the Set of Symbols Occurring in Engineering Mathematics Texts, In *Proc. 8th IAPR International Workshop on Document Analysis Systems, (DAS 2008)*, IEEE Computer Society, pp. 17-19, 2008.

[3] E. Smirnova and S.M. Watt, Context-Sensitive Mathematical Character Recognition, In *Proc. IAPR International Conference on Frontiers in Handwriting Recognition, (ICFHR 2008)*, pp. 604-610, 2008.

[4] B.W. Char and S.M. Watt, Representing and Characterizing Handwritten Mathematical Symbols Through Succinct Functional Approximation, In *Proc. International Conference on Document Analysis and Recognition, (ICDAR)*, IEEE Computer Society, pp. 1198-1202, 2007.

[5] O. Golubitsky and S.M. Watt, Confidence Measures in Recognizing Handwritten Mathematical Symbols, In *Proc. Conferences on Intelligent Computer Mathematics 2009*, Springer Verlag LNCS 5625, pp. 460-466, 2009.

[6] O. Golubitsky and S.M. Watt, Online Computation of Similarity between Handwritten Characters, In *Proc. Document Recognition and Retrieval XVI, (DRR 2009)*, SPIE and IS&T, ISBN 9780819474971, ISSN 0277-786X, pp C1-C10, 2009.

[7] O. Golubitsky and S.M. Watt, Online Stroke Modeling for Handwriting Recognition, In *Proc. 18th Annual International Conference on Computer Science and Software Engineering, (CASCON 2008)*, IBM Canada, ISSN 1705-7345, pp. 72-80, 2008.

[8] O. Golubitsky and S.M. Watt, Distance-Based Classification of Handwritten Symbols, Technical Report, Ontario Research Center for Computer Algebra, `http://www.orcca.on.ca/TechReports/TechReports/2009/TR-09-03.pdf`.

[9] S. Feng, I. Kogan and H. Krim, Classification of Curves in 2D and 3D via Affine Integral Signatures, to appear in *Acta Appl. Math.*, 2008, `http://arxiv.org/abs/0806.1984`.

[10] R. Mukundan, K. R. Ramakrishnan, Moment Functions in Image Analysis: Theory and Applications, Vol. 37, Number 6, pp.15-19, World Scientific, 1998.

[11] M.K. Hu, Visual Pattern Recognition by Moment Invariants, IRE Transactions on Information Theory, Vol. IT-8, pp. 179-187, 1962.

[12] C.L. Liu, K. Nakashima, H. Sako and H. Fujisawa, Handwritten Digit Recognition: Investigation of Normalization and Feature Extraction Techniques, Pattern Recognition, Vol. 37, Issue 2, pp. 265-279, 2004.

[13] M. Mercimek, K. Gulez and T.V. Mumcu, Real Object Recognition Using Moment Invariants, Sadhana, Vol. 37, Number 6, pp.765-775, 2005.

[14] O. Golubitsky and S.M. Watt, Tie Breaking for Curve Multiclassifiers, ORCCA Technical Report TR-09-02, `http://www.orcca.on.ca/TechReports/2009/TR-09-02.pdf` .

[15] Ink Markup Language (InkML) W3C Working Draft, `http://www.w3.org/TR/InkML`, 2006.

Towards context-based disambiguation of mathematical expressions

Mihai Grigore[1] and Magdalena Wolska[2]*
and Michael Kohlhase[1]

[1] School of Engineering & Science, Jacobs University Bremen,
D-28759 Bremen, Germany
{m.grigore,m.kohlhase}@jacobs-university.de

[2] Fachrichtung 4.7 Allgemeine Linguistik, Universität des Saarlandes
66 041 Saarbrücken, Germany
magda@coli.uni-sb.de

Abstract

We present a preliminary study on disambiguation of symbolic expressions in mathematical documents. We propose to use the natural language within which the expressions are embedded to resolve their semantics. The approach is based on establishing a similarity between the expression's discourse context and a set of terms from *Term Clusters* based on OpenMath Content Dictionaries. The Term Clusters are semi-automatically constructed terminological resources which classify related mathematical concepts into groups. Each group is labelled with a term which represents the common denominator between the concepts.

1 Motivation

Technical and scientific documents have been gaining increasing attention in the computational linguistics community. These documents stretch the current natural language processing technology, among others because they contain embedded structures such as tables, diagrams, or mathematical formulae, which interact with the textual content. While interpretation of such structures is currently outside of the state-of-the-art in language processing, their automatic understanding will enable us to provide services such as fact search, plagiarism detection, and change management for technical and scientific documents. In this paper we address this problem from a linguistic perspective and present a step towards the semantics construction of mathematical formulae. Concretely, we address symbol overloading as one of the sources of ambiguities occurring in mathematical notations.

Authors tend to exploit established conventions in mathematical notation leaving some of the ambiguous notation without explicit explanation, relying on the reader being able to recover the intended meaning. Consider the expression "ω^{-1}": If ω is known to be a function, then ω^{-1} is the inverse function corresponding to ω. However, if ω is a scalar, ω^{-1} should be understood as $1/\omega$. Consider now the following expression: "$S^{-B}f(C\ln S)$". It is a complex term consisting of two subterms, S^{-B} and f, whose concatenation denotes multiplication: S^{-B} is a scalar by which f is multiplied. $f(C\ln S)$, in turn, is a function: here, concatenation of f and $(C\ln S)$ denotes function application. The different appropriate interpretations of the superscript and symbol concatenation are taken for granted by the reader.

*Correspondence to: Universität des Saarlandes, Allgemeine Linguistik (FR 4.7), Postfach 15 11 50, 66 041 Saarbrücken, Germany; Tel:+49-681-3024345 Fax:+49-681-3024351

Mathematical discourse, however, does not consist of symbolic expressions alone. It is rather the familiar combination of natural language and symbolic expressions. Crucially for interpretation, the expressions' linguistic context often contains information which helps determine the expressions' meaning. Readers can therefore in many cases immediately resolve the intended reading of mathematical expressions by looking at the embedding discourse context. For instance, in the second example the text actually reads:

> *"... The scaling function has the form $S^{-B} f(C \ln S)$, where f is a 2π-periodic function."* from [9]

Given this linguistic context, the symbolic expression can be immediately interpreted to denote a function and with the knowledge that S^{-B} is a scalar, the internal structure of the expression can be identified.

The intended interpretation of symbolic mathematical expressions can be a useful source of information in a number of sub-tasks in a mathematical document processing pipeline for digitalising mathematics. For instance, in the task of parsing mathematical notation, i.e. identifying the structure and (compositional) semantics of symbolic expressions, the information about the expressions' interpretation can guide the selection (or weighing) of likely parse candidates. This could be useful in processing LaTeX documents as well as in mathematical OCR, in particular, in handwriting recognition; for instance, in examples such as above, in deciding between horizontal adjacency and super-/subscript relation when the super-/subscript is written partly across the centre horizontal line of the expression.

In this paper we present a preliminary study on disambiguating a certain subset of symbolic expressions in mathematical documents. Namely, we focus on those mathematical expressions which are syntactically part of a *nominal group* and, in particular, are in an *apposition relation* with an immediately preceding noun phrase. That is, our target expressions come from a linguistic pattern: "*... noun_phrase symbolic_math_expression ...*" (as in the example above). Therefore, in the approach described here we assume that a target mathematical expression can be disambiguated using its *left* context.

We formulate the disambiguation problem as follows: Given a mathematical document containing a target mathematical expression, can we indicate one (or more) concepts from a predefined set of concepts as the interpretation of the given expression. We claim that the linguistic context information improves disambiguation accuracy.

Our approach is based on the use of natural language lexical context information which is contained in the natural language surrounding a target expression. We compute a semantic similarity between the words from the lexical context of a given expression and a set of terms from manually constructed and semi-automatically extended *Term Clusters* based on OpenMath. Lexical contexts for the Term Clusters are compiled from a large corpus. As interpretation of a mathematical expression we consider the cluster (represented by its name) with the highest similarity to the words in the context.

Outline The paper is organised as follows: In Section 2 we describe our approach to disambiguation of appositional mathematical expressions: the lexical resources it uses and the word similarity-based disambiguation. In Section 3 we outline the experiment setup: the data we used, the performance metrics, and the baselines. The results are presented in Section 4. In Section 5 we present conclusions and discuss further work.

Symbol Name	CD Name (CD Group)	Description
`inverse`	`fns1` (Functional Operators)	This symbol is used to describe the inverse of its argument (a function). This inverse may only be partially defined because the function may not have been surjective. If the function is not surjective the inverse function is ill-defined without further stipulations. No assumptions are made on the semantics of this inverse.
`inverse`	`arith2` (Arithmetic Functions)	A unary operator which represents the inverse of an element of a set. This symbol could be used to represent additive or multiplicative inverses.
`eigenvector`	`linalg4` (Linear Algebra)	This symbol represents the eigenvector of a matrix. It takes two arguments the first should be the matrix, the second should be an index to specify which eigenvalue this eigenvector should be paired with. The ordering is as given in the eigenvalue symbol. A definition of eigenvector is given in Elementary Linear Algebra, Stanley I. Grossman in Definition 1 of chapter 6, page 533.

Table 1: Excerpt from OpenMath Content Dictionaries

2 The Approach

Our approach is based on two observations: First, the linguistic context is often a good indicator of the intended semantics of a mathematical expression. Second, similar lexical contexts indicate similar mathematical domain content. The central part of the approach is the use of co-occurrence statistics in computing semantic similarity between the words from the mathematical expressions' context and a set of terms from predefined semantic classes. These are represented by manually constructed Term Clusters which model certain logical classes of mathematical concepts according to lexical terms which evoke them.

2.1 Lexical Resources

The main resource on which our approach relies is a collection of *Term Clusters* (TC) which we derive from OpenMath *Content Dictionaries* (CDs). OpenMath [6, 14] is a language for representing mathematics, in particular symbolic mathematical expressions, both at the surface structure and the semantics level. It is becoming a de facto standard for communicating content-based mathematics over the Web. OpenMath uses CDs to define the semantics of symbols used to build mathematical expressions.

OpenMath Content Dictionaries The OpenMath CDs group symbol definitions by sub-fields of mathematics and carry names (e.g. arith2, linalg4, set1) that reflect this (see Table 1). We make use of the fact that they are further organized into groups according to mathematical areas (Arithmetic Functions, Linear Algebra, etc.). Note that mathematical concepts are identified by a symbol name (e.g. inverse, eigenvector, emptyset) together with CD names, so that CDs can have symbols of the same name, which are nonetheless different mathematical concepts. Note also that one CD can belong to more than one group.

TCN	Representative Terms
`algebraic structure`	algebra, array, basis, field, generator, group, groupoid, ideal, lattice, matroid, monoid, quaternion, ring, semigroup, space, subfield, submonoid, subsemigroup, ...
`function`	application, automorphism, closure, convolution, density, eigenfunction, endomorphism, entropy, function, functor, hamiltonian, hermitian, homomorphism, homotropy, inverse, isomorphism, lagrangian, logarithm, map, morphism, transformation, ...
`property`	associativity, commutativity, concavity, continuity, convexity, differentiability, diffusion, distributivity, goodness, linearity, noise, property, violation, ...

Table 2: Excerpt of the Term Clusters

Each symbol declaration in a CD is accompanied by a natural language description of the symbol's meaning. The descriptions are written according to an informal set of guidelines [8]. They share a certain formulaic style, relatively simple syntactic structure, and are characterized by brevity. There are 190 CDs in the currently available OpenMath [15]. Table 1 shows the definitions of an *eigenvector* and two definitions of the *inverse* concepts.

Term Clusters for Mathematical Expression Disambiguation We will use the OpenMath CDs to build Term Clusters as linguistic terminological resources that group mathematical lexical terms (corresponding to mathematical concepts) into sets of terms subordinate to a more general term (concept). It is these more general terms, Term Cluster Names (TCN), that we will use as interpretations of mathematical expressions in the disambiguation process: for each target expression we will assign a TCN as its interpretation.

For the experiments described in this paper we created 17 TCs semi-automatically. The structure and the content of the TCs were to a large extent inspired and semi-automatically extracted from the OpenMath Content Dictionaries.* The Term Clusters were constructed as follows: First, we extracted mathematical terms from each OpenMath CD and removed modifiers from multi-word terms obtaining bare nouns. Next, the CDs associated with the same mathematical area were collapsed (e.g. `arith1`, `arith2` and `arith3` correspond to `arithmetics`). The same procedure was applied to pairs of CDs with high term overlaps. We then extended the set of terms per cluster by extracting the top 100 most frequently co-occurring words from 10,000 documents from the arXMLiv collection [17]. Co-occurrence frequencies were calculated for pairs of words within one sentence.

Finally, in order to further enrich the lexical resource we partially automatically and partially manually extracted terms from various online lexica of mathematical terms, in particular, the University of Cambridge mathematical thesaurus [7] and the MathWorld lexicon of mathematical terms [11], and added them to the appropriate Term Clusters. The lexical entries in the TCs are bare nouns in the singular form. Table 2 shows excerpts from the TCs *algebraic structure*, *function* and *property*.

*In initial experiments, we used a pre-processed version of OpenMath CDs , but found its granularity too fine-grained on the one hand, and on the other hand, its coverage too limited.

2.2 Mathematical Expression Disambiguation

The method we propose is inspired by recent computational approaches to word sense disambiguation and lexical similarity which use statistical association measures to estimate semantic relatedness between words by analysing their distributional properties. In these approaches distributional similarity is computed based on, for instance, WordNet glosses [21] the Wikipedia, large corpora, or even the Web (see, for instance, [3, 12, 13, 16, 19]).

By analogy, our approach considers similarity between the given mathematical expression's lexical context and the terms in Term Clusters introduced above. The disambiguation process consists of three parts: First, we preprocess the documents and identify the candidate target mathematical expressions, then we compute corpus-based similarities for each TC, and finally disambiguate each target expression based on the words in their context and the TCs. Below we briefly present the disambiguation components.

Preprocessing We used 10,000 mathematical documents from the arXMLiv collection [17], word- and sentence-tokenized them, stemmed the words, and normalized the mathematical expressions by replacing them with a unique identifier. This preprocessing was performed both on the corpus for similarity computation and on the documents we used for evaluation of the approach. The latter documents were also processed with the Stanford part-of-speech (POS) tagger [18] in order to identify nouns in the left context of appositional mathematical expressions. The recall results we report in Section 4 are based on the output of this tagger.[†]

The evaluation set we furthermore processed in two ways in order to obtain results for alternative approaches to selection of candidates for computing semantic similarity: stop-word based and POS-based. In the stop-word approach the candidate terms for computing similarity were selected using a stop-word list alone. In the POS approach we used those words which were tagged as nouns by the Stanford tagger.

Computing term similarity We experimented with three statistics to find words semantically related to the TC terms: the Dice coefficient (*Dice*), the pointwise mutual information (*PMI*) and the z-score (z). These measures estimate relative probability with which words occur in proximity and have been previously successfully used in computational linguistics [4, 5, 20]. For two words w_1 and w_2, *Dice* is defined as twice the ratio of the joint probability to the sum of the individual probabilities, *PMI* is defined as the *log* of the ratio of the probability of the words occurring together to the product of the individual probabilities, and z is the proportion of the difference of the expected and the observed probabilities (P) to the expected probability (E):

$$Dice(w_1, w_2) = \frac{2 \times P(w_1, w_2)}{P(w_1) + P(w_2)}$$

$$PMI(w_1, w_2) = \log_2 \frac{P(w_1,w_2)}{P(w_1)\times P(w_2)} \qquad z(w_1, w_2) = \frac{P(w_1,w_2)-E(w_1,w_2)}{\sqrt{E(w_1,w_2)}}$$

Because *log* is monotonically increasing, relative ordinal rankings of *PMI* estimates are preserved if *log* is dropped. We approximate the probabilities by raw frequencies, as is a common practice.

[†]We are aware of the fact that these results are affected by the low performance of the tagger which was not trained on documents from our domain. To our knowledge, there are currently no available dedicated language processing tools, in particular, part-of-speech taggers, for mathematical discourse. In the future, we are planning to develop a dedicated tagger and re-evaluate our approach on more accurate POS outputs.

Based on experimentation we used the *Dice* coefficient for those words with *Dice* scores higher than a certain predefined threshold, $\lambda \approx 0.6$, otherwise we used either *PMI* or z:

$$sim(w_1, w_2) = \begin{cases} Dice(w_1, w_2) & \text{if } Dice(w_1, w_2) > \lambda \\ PMI(w_1, w_2) \text{ or } z(w_1, w_2) & \text{otherwise} \end{cases}$$

PMI is a frequently used statistical estimator of the strength of word co-occurrence; see, for example, [20]. The use of different lexical co-occurrence statistics is motivated by the fact that mutual information is reported to be less efficient on low-frequency events [10]. Comparisons with latent semantic analysis show that *PMI* achieves better results when large amounts of data is used [4, 5]. In Section 4 we report results for different similarity thresholds: for *PMI* we experimented with thresholds $\delta \in \{0.6, 0.8, 0.9\}$, while for z the considered values were $\delta \in \{0.0, 10.0, 20.0\}$ Lexical similarity was computed based on a subset of arXMLiv documents preprocessed as described above. The obtained co-occurrence pairs were type- not token-based, i.e. they contained only word-stems.

Disambiguation For each mathematical expression identified as appositional (i.e. preceded by a noun, based on the output of the Stanford parser; see Section 3 for the details on the evaluation set) we considered a local context C consisting of all the nouns appearing in the five word window to the left of a target mathematical expression. For each candidate noun w in the context C we identified the TC terms, *tct*, with the highest semantic similarity according to the similarity metrics described above. In order to identify the TC which best matches the context, we used modified versions of similarity measures presented in [12]. The obtained similarity scores were weighted, summed up, and normalized by the length of the considered context (e.g. the number of nouns found within the five word windows). In weighing the candidates we took into account the distance to the target expression; with the weights decreasing with the distance to the target expression. The similarity was calculated using the following scoring function:

$$Sim(C, TC) = \sum_{w \in C} maxsim(w, TC) \times cw(w), \text{ where}$$

$$maxsim(w, TC) = \max_{tct \in TC} \{sim(w, tct)\} \text{ and } cw(w) \text{ is the weight for the word } w$$

That is, the resulting assigned interpretation is the TC with the highest similarity score between the lexical context and the terms from each of the sets of TC terms.

3 Experiment Setup

We tested the disambiguation method on a manually constructed gold-standard, a set of manually identified and disambiguated appositive mathematical expressions. We compared the algorithm's performance with two baselines which do not use context or have access to limited context information. Below we introduce our evaluation sets, define the performance measures we employed and present the baselines.

Data We conducted an initial evaluation of the approach on all mathematical expressions from one randomly selected document from the arXMLiv containing 451 mathematical expressions [2]; we will refer to this initial evaluation set as *init-set*.

In order to obtain more reliable performance results, in particular, on disambiguation in documents originating from various authors and mathematical sub-areas, we also conducted further evaluation on a set of randomly selected mathematical expressions extracted from a random collection of different documents. This evaluation set (*eval-set*) was constructed as follows: First, we selected 28 random arXMLiv documents which were successfully preprocessed. Second, from each of these documents we selected 20 random mathematical expressions and manually identified the appositive cases among those, obtaining 116 appositive instances. Third, we manually annotated this set with the expected categories, thereby creating a *gold standard*. The gold standard contains 101 disambiguated appositive mathematical expressions. (15 unclear cases from the original set were discarded.)

Performance measures As evaluation metrics we use precision (P), recall (R), $F_{0.5}$, and Mean Reciprocal Rank (MRR). Precision and recall are set-based measures. In classification, precision is the proportion of correctly labelled examples, while recall is the proportion of labelled examples out of all examples. In mathematical expression disambiguation we prefer correct disambiguation over coverage, therefore, we choose $F_{0.5}$ as a combined measure. $F_{0.5}$ is a variant of the harmonic mean of precision and recall which weights precision twice as high as recall. MRR is one of the standard measures used in Information Retrieval for evaluating performance of systems which produce ranked lists of results, for example, ordered lists of documents retrieved in response to a query. It is the inverse of the rank of the expected (best) result item. More specifically,

$$P = \frac{tp}{tp+fp} \quad R = \frac{tp}{tp+fn} \quad F_{0.5} = \frac{(0.5^2+1)PR}{0.5^2P+R} \qquad MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}$$

where *tp* are true positive classifications, *fp* are false positives, *fn* are false negatives, and N is the number of evaluated instances.

Baselines We employed two baselines for comparison with our approach. The trivial baseline does not use any context information and simply assigns a random order of categories. The top random category is used in calculating precision. We can consider this as the lower-bound for the performance of the approach. The second baseline uses limited context information: It uses only the noun (*NN*) immediately preceding the target mathematical expression as candidate for disambiguation.

4 Results

Table 3 summarises the results of the evaluation. *Init-set* is our initial set of mathematical expressions from a single document. *Eval-set* is our gold-standard evaluation set. Results are reported for the two baselines: random ranking (*random*) and limited context (*nearest NN*). SW is the approach which uses stop-words to select candidates for similarity computation and *PMI* as the similarity measure. POS-z and POS-*PMI* are the approaches which use POS tags for candidate selection and z and *PMI* as co-occurrence statistics alternative to Dice. δ are the different similarity thresholds.

Init-set was our preliminary evaluation set which served to verify the plausibility of the approach. We did not calculate MRR for this set, however, with the average precision at 81% (for $\delta = 0.9$) we considered the approach promising. With limited access to context information both baselines perform poorly, as expected; recall is not reported because for the baselines some interpretation is always returned. In future experiments we will use

			δ	P	R	$F_{0.5}$	MRR
Init-set	POS-*PMI*		0.6	64.00	31.00	52.77	–
			0.9	81.00	21.00	51.55	–
Eval-set	Baselines	*random*	–	8.51	–	–	0.19
		nearest NN	–	20.21	–	–	0.32
	SW		0.6	53.09	81.13	57.03	0.60
			0.8	61.36	36.49	54.00	0.68
			0.9	63.16	14.28	37.50	0.74
	POS-z		0.0	56.96	44.55	53.96	0.68
			10.0	65.52	37.62	57.06	0.74
			20.0	69.70	22.77	49.36	0.77
	POS-*PMI*		0.6	65.67	43.56	59.62	0.76
			0.8	80.39	40.59	67.21	0.85
			0.9	83.33	39.60	68.26	0.87

Table 3: Evaluation results

other baselines with less limited context information, but limited capabilities of similarity estimation; one plausible baseline could use larger context (e.g. five word window) but calculate only word overlap with Term Clusters, rather than corpus-based similarity.

Considering the limited linguistic preprocessing we employ (the method is based solely on co-occurrence statistics with only stemming, stop-words and largely faulty POS tagging as preprocessing) both the precision results and the ranking results on the evaluation set are encouraging. *PMI* appears to outperform the z-score on this task. Interestingly, the results of the best performing stop-word based model appear comparable not only with the z-score models, but also with the *PMI*-based model at $\delta = 0.6$. This suggests that perhaps further work could be also invested in the knowledge-poor approaches, given the lack of reliable language processing tools for mathematical discourse. Moreover, not surprisingly, with all the measures, the performance is strongly sensitive to the similarity thresholds. We must perform further systematic analyses of the effect of different threshold combinations.

5 Conclusion and Future Work

We can cautiously conclude that the method produces promising results and that, even with limited linguistic information, the lexical context provides useful information in mathematical expression disambiguation. Of course more work and experimentation is needed to further tune the co-occurrence statistics and the similarity metrics. In particular, we are planning to experiment with other corpus-based term association measures. Moreover, we have started to experiment with methods analogous to those presented here, but applied to mathematical expressions which need the *right* context for disambiguation.

Our Term Clusters require further work. In their present state some of the clusters group unrelated terms; see, for instance, the terms grouped under *property*. We are currently working on a more coherent resource with a richer hierarchical structure and with thesaurus-like relations between concepts. With such a resource we can investigate thesaurus-based similarities based on relations such as "broader/narrower concept". As an initial step we are planning to investigate the relations included in the Cambridge Mathematical Thesaurus.

Similarity could be computed as inversely proportional to the distance between words in the thesaurus hierarchy; short paths between two concepts would indicate high degree of semantic similarity. This is analogous to the way WordNet is used in lexical similarity tasks.

Another resource that we plan to take into account is the "Mathematics Subject Classification" (MSC [1]), a hierarchically organized set of over 5000 mathematical subjects. These could act as Term Cluster Names that cover all of mathematics. We will try to generate the representative terms from the Zentralblatt Math corpus [22], an MSC-classified set of 2.5 million abstracts of mathematical (journal) publications of the last 100 years. We expect to obtain much more accurate disambiguation results from such a resource.

It is clear that linguistic knowledge would help in the disambiguation task. For the left-context appositional cases, the noun phrase part of the nominal group is alone sufficient for disambiguation. However, in order to be able to perform more linguistically-based analysis of the context, we need language processing tools (POS taggers, chunkers, etc.) which are nowadays taken for granted in language processing. Unfortunately, existing tools are typically trained on newspaper text and therefore produce sub-standard results on the mathematical genre. A serious problem here is the lack of annotated data to re-train such tools. We are presently investigating ways of creating a POS-annotated document set and building a specialised POS tagger.

Finally note that our definition of disambiguation still falls significantly short of semantics construction for formulae, where every symbol is interpreted by a semantic concept. Our approach based on the left context and general mathematical resources cannot work since mathematical texts are well-known to introduce notations and concepts as they go along. We would need a deeper discourse analysis that detects notation introductions (and imports for that matter) and brings them to bear locally in the disambiguation process.

Acknowledgements

We would like to thank Deyan Ginev of Jacobs University Bremen without whose many preprocessing scripts it would not have been possible to conduct this study at ease. We would also like to thank three anonymous reviewers for helpful comments.

References and Notes

[1] American Mathematical Society. 2010 Mathematics Subject Classification. `http://www.ams.org/mathscinet/msc/`, 2009.

[2] R. Bañuelos, T. Kulczycki, and P. J. Méndez-Hernández. On the shape of the ground state eigenfunction for stable processes. *Potential Analysis*, 24(2):205–221, 2006. `http://arxiv.org/abs/math/0407260`; Retrieved October 2009.

[3] D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *Proceedings of the 16th international conference on World Wide Web*, pages 757–766, 2007.

[4] R. Budiu, C. Royer, and P. Pirolli. Modeling information scent: a comparison of LSA, PMI-IR and GLSA similarity measures on common tests and corpora. In *Proceedings of the 8th RIAO Conference*, 2007.

[5] J. Bullinaria and J. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39(3):510–526, 2007.

[6] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaetano, and M. Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004.

[7] The University of Cambridge Mathematical Thesaurus. `http://www.openmath.org`, seen October 2009.

[8] J. Davenport. On writing OpenMath content dictionaries. Technical report, The OpenMath Esprit Project, 2002. `http://www.openmath.org/documents/writingCDs.pdf`.

[9] F.-J. Elmer. Self-organized criticality in the weakly driven Frenkel-Kontorova model. *Physical Review E (Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics)*, 56(6):R6225–R6228, 1997. `http://arxiv.org/abs/adap-org/9710002`; Retrieved October 2009.

[10] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. Massachusetts Institute of Technology, Revised version May 1999, 2003.

[11] MathWorld. `http://mathworld.wolfram.com/`, October 2009.

[12] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 775–780. AAAI Press, 2006.

[13] S. Mohammad and G. Hirst. Determining word sense dominance using a thesaurus. In *Proceedings of the 11st Conference of the European Chapter of the Association for Computational Linguistics*, pages 121–128, 2006.

[14] OpenMath. `http://wiki.openmath.org`, October 2009.

[15] OpenMath Content Dictionaries. `http://wiki.openmath.org/cdnames`, October 2009.

[16] S. Patwardhan, S. Banerjee, and T. Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the 4th International Conference on Computational Linguistics and Intellignet Text Processing*, pages 241–257, 2003.

[17] H. Stamerjohanns, M. Kohlhase, D. Ginev, C. David, and B. Miller. Transforming large collections of scientific publications to XML. *Mathematics in Computer Science*, 2009. in press, see `http://kwarc.info/kohlhase/papers/mcs09.pdf`.

[18] Stanford Log-linear Part-Of-Speech Tagger. `http://nlp.stanford.edu/software/tagger.shtml`, October 2009.

[19] M. Strube and S. P. Ponzetto. WikiRelate! computing semantic relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1419–1424, 2006.

[20] P. D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. *Lecture Notes in Computer Science*, pages 491–502, 2001.

[21] WordNet. `http://wordnet.princeton.edu`, October 2009.

[22] Zentralblatt MATH. `http://www.zentralblatt-math.org`, October 2009.

Digitisation Workflow in the Czech Digital Mathematics Library

PETR SOJKA

Faculty of Informatics, Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
sojka@fi.muni.cz

Abstract

Experience in setting up a workflow from scanned images of mathematical writings into a fully fledged mathematical library is described on the example of the project Czech Digital Mathematics Library DML-CZ. An overview of the whole process is given, with detailed description of production steps involving scanned image processing and optical character recognition. Experience gained, lessons learned and tools prepared during development of DML-CZ are described. DML-CZ now serves over 25,600 articles (275,000 digitised pages) to the public.

Keywords: digital mathematical library, mathematical knowledge representation, digitisation workflow, optical character recognition, OCR, retro-digitisation, DML-CZ

Viva la Workflows! (Carole Goble [6])

1 Motivation

Digital Library business has moved from data/files centered processing towards process-oriented *workflows*. Workflows enact the machinery of building and running a digital library. Instead of running simple tools and mirroring file repositories more subtle solutions have to be devised: data curatorship changes to workflow curatorship and *services*.

There are communities and systems that start to dominate in some thematic areas: PubMed Central (PMC) is one such system in the medical domain, speeding up research and author's citation indexes in the area. Unfortunately, only domains where global initial funding was available took advantages of the platforms established and tools and workflows developed. In the PMC case, journal publishers are now eager to join the club, and authors enjoy global topical ontology-based search. Researchers send their papers only to journals available in PMC as this leverages their citation indexes. However, the realization of the dream of a World Digital Mathematics Library [7] is yet to come.

We report on the experience gained, lessons learned and tools prepared during the development of a digitisation workflow for The Czech Digital Mathematics Library DML-CZ project. The aim of the project approved for the five years period 2005–2009 was to digitize the relevant mathematical literature published in the Czech lands. It comprises periodicals, selected monographs and conference proceedings from the nineteenth century up until currently produced mathematical publications. It has been launched and is readily available on dml.cz, serving more than 25,600 articles on 275,000 pages to the public.

The general workflow of the project, shown on Figure 1 on the following page, reflects different types of acquired input data:

full digitisation from print work starts from a paper copy;

full digitisation from bitmap image work starts from an electronic bitmap of pages;

retro-born-digital work starts from an electronic version of the document (usually in POSTSCRIPT or PDF);

born-digital workflow of the journal production is enriched with an automated export of data for the digital library.

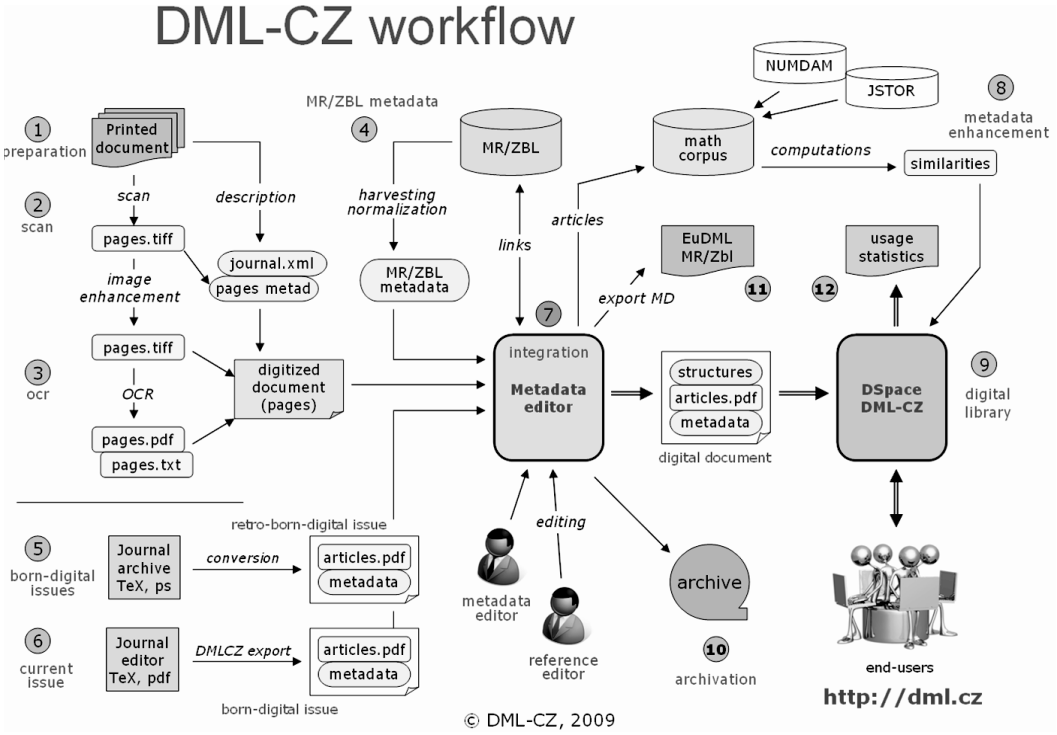


Figure 1: DML-CZ top-level workflow scheme

Within the project, several general purpose tools have been developed:

1. scripting of transformation pipes of scanned images,
2. DML-CZ OCR workflow allowing recognition of scanned mathematical documents,
3. web-based Metadata Editor [1],
4. tools for classification of mathematical documents and measuring their similarity [14];
5. workflow for born-digital publication production with direct export of metadata for DML [15] and
6. plenty of other smaller tools like: extensions to LUCENE engine allowing indexing of mathematics, batch PDF stamper for digital signing of produced PDF, an optimizer recompressing image objects in PDF with the new JBIG compression filter supported by Adobe since PDF specification version 1.6 (Adobe Reader 5) or batch article PDF generation with titlepage by XeLaTeX.

In the following sections we describe part (steps 2, 3, 7 and 8 in Figure 1) of our digitisation workflow with the hope that they can be used by similar projects or even in other domains.

We are all apprentices in a craft where no-one ever becomes a master. (Ernest Hemingway)

2 Scanning and Image Transformations

Processing of scanned images is aimed at final delivery of 600 DPI bi-tonal images suitable for quality OCR and a fine print. This is the quality recommended by the Committee on Electronic Information and Communication (CEIC) and used for example by JSTOR and NUMDAM. Images from the Göttingen Digitisation Centre (GDZ) and images scanned in the Digitisation Centre of the Library of Academy of Sciences, Czech Republic prior to the project DML-CZ have bi-tonal 400 DPI quality. The difference is visible, and leads to a higher OCR error rate. We strongly support the recommendation to scan with a resolution of at least 600 DPI.

We perform our new scans at 600 DPI with 4-bit depth, 'having a depth/space' for geometrical and other transformations done on images before binarization. Scanning is carried out in Digitisation Centre of the Library of Academy of Sciences in Jenštejn near Prague on the Zeutschel OS 7000 A2 book scanners.

The primary scans in TIFF format are archived for a possible future reprocessing if needed. We use BOOK RESTORER™ image restoration software by i2S for interactive and batch image processing in an uncompressed TIFF format. Operations performed on images are:

1. *geometrical correction* as narrowing the baselines and widths of the same characters on the same line;
2. *cropping* the page to cut out the speckles at page borders;
3. *blur filter*, 3×3 pixels, to eliminate one or two pixel size variations;
4. *binarisation* with manually adjusted parameters for every batch (usually journal volume);
5. *despeckle filter*, with both white and black spotting, 3×3 pixels;
6. *publish/export*: processed TIFFs are stored being compressed by the Lempel-Ziv-Welsh method for compressing grayscale and the G4 one for binarized images to speed up further processing (OCR) and to save space.

Both the order of these steps and the parameter adjustments for images of different quality are very important. For the data from GDZ, slightly different operations are needed as the input files are already bi-tonal and some filters are applicable only on grayscale images.

Step 1 employs the algorithms that allow perspective correction of a scanned image. As most of the material to digitize cannot be cut, we scan 2-up page spreads, making the text size non-uniform even when trying to flatten the spread by pane of glass. Book Restorer can also flatten the lighting across the scanned spead. For more details of this step see [2, page 2].

Step 2 crops the unnecessary border parts of the page shot.

Step 3 aims at better binarisation and despeckling by unsharping the shapes in the image.

Step 4 is necessary as most OCR engines work on bi-tonal images. It may be left to the high-quality OCR engine—clever thresholding starts to be a standard part of OCR programs [18], or perform it ourselves adaptively based on OCR feedback [13].

Step 5 is inserted to remove small impurities in the image.

Step 6 is the final step: image is stored as LZW-compressed grayscale or G4-compressed bi-tonal TIFF.

For the lower resolution data from GDZ, slightly different operations are needed as the input files are already bi-tonal (e.g. we did upscaling before unsharping) and because some filters are applicable only on grayscale images.

It is wise to differentiate processing of pages with grayscale images (e.g. photos) so that they are not degraded by image filters suitable for text. To avoid possible difficulties in the later steps it is important from the very beginning to carefully check image quality before proceeding with the remaining steps. At least automated procedures that check the technical metadata (e.g. `tiffinfo`) and image quality (pixel width and height) has to be the part of quality assurance. Metrics of compressibility by the JBIG2 encoder were used to trigger quality checks.

There is a tradeoff between price of image cleanup and quality results within constraints of digitisation budget. When acquiring a craftmanship in good image editing software, results very close to (or even better than) the original could be achieved [17]. These hand made touches are usually beyond the budget of most digitisation projects, where the highest degree of automation is needed to reduce the cost of digitisation. In DML-CZ, we have prepared [12] set of batches of typical transformation procedures to be used by BOOK RESTORER™ operators to achieve the best price/effort ratio.

Fine-tuning of operations on the pixel level pays back in the following step: the OCR.

The road to wisdom?
Well, it's plain and simple to express:
Err and err and err again,
but less and less and less. (Piet Hein)

3 Optical Character Recognition – DML-CZ OCR

To have papers indexed we need to get full text from page bitmaps by the process of optical character recognition. Also, we need to recognize logical page numbers located in every TIFF, to link the page images to article metadata.

Tests with various OCR programmes showed that no single one gives acceptable results for mathematical content, with character error rates often above 10% (counting wrong character positions and font types as errors too). For text recognition, FINEREADER by ABBYY® gave the best results, whereas for the structural recognition of mathematics InftyReader [24] had impressive results.

The FINEREADER software development kit (SDK for Windows version 8.1) was used to develop a part of the system for the location and recognition of page numbers, and a batch system DML-CZ OCR [19, 22] which takes sequences of TIFF images and produces two-layered one page PDFs (with invisible full-texts behind the images). The processing starts with the recognition of languages used in every paragraph, and then blocks are recognized again with a special setting (language dictionaries used) for every given block of text. With such a fine-tuning of parameters, we are able to achieve a one percent character error rate [22].

Among solutions and software evaluated on plain texts, FINEREADER gave the best results, but it has no support for the recognition of mathematical expressions. Texts without recognized maths may be sufficient for basic indexing and search. However, it is not surprising that omitting maths matters when the full texts are used for such tasks as automated text classification and categorization or for computing paper similarity [23]. Therefore we strive to enhance the state-of-the-art possibilities for mathematical OCR.

Neither ABBYY® nor Google responded positively on the near future of math OCR development plans—mathematics is only a small market niche for them. On the other hand, developers of the INFTYREADER system [24] were willing to gradually improve their support for European languages, MATHML and LaTeX export filters and to enrich their recognized database of mathematical symbols.

We found that setting the parameters of the OCR engine (language, word-list consultation) influences the precision significantly. We trained FINEREADER on the type cases used at the printer where journals were typeset.

At the end of extensive experiments, we developed a method of OCR processing consisting of several phases, both in FINEREADER and Infty. Processing using FINEREADER consist of the following:

1. A page or block of text is recognised for the first time using a universal setup (non-language specific). A histogram of character bigrams and trigrams from words with lengths greater than three is created.

2. The computed histogram of the text block is compared [5] to the histograms created from the journal data during the training phase for all languages used (English, French, Russian, German and Czech). Perl module `Lingua::Ident` is used. Block with bibliography is detected by different algorithms and is treated differently.

3. Page or block of text is processed for the second time with parameters optimised for recognised 'language' in previous step and saved as a two-layer PDF (with text layer used for searching, indexing and similarity computation).

Recognition of mathematical formulae in FINEREADER is not satisfactory, however. The only suitable tool for this domain that we have found and experimented with is INFTY. INFTY's new PDF import capability is very significant to us: it will allow to import our current FINEREADER's two-layer PDFs, use the text part only, throw away badly recognized maths and to detect and recognize maths expressions. A new INFTY version that combines FINEREADER's technology (OCR voting [9]) is in preparation. In the meantime,

1. PDF is passed to INFTYREADER and results are stored in the INFTY Markup Language (IML) and in LaTeX (Human readable LaTeX).

2. IML is postprocessed by a home-grown programme in JAVA to fix recognition errors of some of the accented characters that INFTY does not yet have in its glyph database.

Using the process outlined above we have managed to decrease the character error rate from an initial 11.35% (universal language setup of FineReader) to an average 0.98% character error rate. [10, 11, 20] The whole processing is fully automated after initial font recognition and language detection training. The error rate may be further decreased when INFTY's character database is semiautomatically enriched when processing a new journal.

When in doubt, use brute force. (Ken Thompson)

4 Text Postprocessing and Metadata Enhancements

The OCR step is followed by further text processing, and its results are used for editing of metadata and references.

4.1 Metadata Editor

The Metadata Editor (ME) [1, 4] has gradually developed into a fully-fledged and efficient web application, `https://editor.dml.cz`, that allows simultaneous remote editing according to assigned structured access rights. It supports two levels of actions. On the first one the operator editing the data is provided with page thumbnails so that he can visually check the completeness, scan the quality and configuration of the articles, easily shuffle the pages and cut or merge articles if necessary. On the other level the operator can check the automatically imported metadata, edit and complete them. An integral part of the ME is the module for administration of authority files with authors' names. It enables the most suitable version of the name for the DML-CZ to be selected and to match it with all its other versions.

We consider bibliographical references as important metadata of every paper. Their availability makes it possible to use professional systems like CrossRef® for cross-publisher citation linking. The work starts from OCR of the text, in which a block of references is found. Citations are tagged by a script based on regular expressions written for the citation style of every journal. The operator then checks, edits and approves the list of paper citations.

For fixing errors that can be safely detected (such as a Mathematics Subject Classification (MSC) code string that is invalid in the MSC 2000 standard) procedures are formulated and coded in XSchema generated also from a web-based interface (forms). Other sets of constraint checkers run as overnight jobs together with updates of the database and metadata statistics and logs useful for the management of Metadata Editor workflow.

Finally, various detection procedures for possible errors have been suggested, evaluated and implemented for finding anomalous and suspicious content of metadata fields, with lists of warnings generated, including hyperlinks for easy checking by an operator. An important control concerns the integrity of TEX sequences in metadata to assure seamless typesetting of article cover pages in the later stages: all metadata to be typeset are exported in one big file with unique references to the article, and typeset by XeLaTeX to check the TEX control sequences used in the metadata fields. This ensures that all of the TEX encoded mathematics converts into MathML format smoothly. Similar procedures allow for an efficient and economical increase of metadata completeness and quality.

4.2 Mathematical Document Classification and Categorization

Article full texts have many applications, e.g. for document classification and categorization. Fine document classification allows document filtering to reach higher precision in information retrieval systems such as DML. The most commonly used classification system today is the Mathematics Subject Classification (MSC) scheme (www.ams.org/msc/), We have developed an MSC classifier (guessed MSC) that is able to assign top-level MSC for retro-digitized articles. Our results convincingly demonstrated the feasibility of a machine learning approach to the classification of mathematical papers [14].

Another round of experiments was done with mathematical document similarity computation. We have collected corpus of full texts of more than 40,000 articles (from DML-CZ and NUMDAM)

and we have computed paper similarities using *tfidf* [16] and Latent Semantic Analysis (LSA) [3] and Random Projection methods. Methods use a Vector Space Model, first converting articles to vectors and then using the cosine of the angle between the two document vectors to assess their content similarity [8]. The difference between the methods is that while *tfidf* works directly over tokens, LSA first extracts concepts, then projects the vectors into this conceptual space where it only computes similarity.

We are now showing the links to closest document lists in DML-CZ article landing pages to get feedback from authors and readers to evaluate metrics computed in this experiment. Given that we will enrich our full text mathematical corpus significantly (with data from JSTOR, ARXIV and other sources as planned), we hope it will help to tackle plagiarism, too.

> Automating the creation of useful digital libraries—that is, digital libraries affording searchable text and reusable output—is a complicated process, whether the original library is paper-based or already available in electronic form. (Simske and Lin [17])

5 Summary, Conclusions and Acknowledgement

We have described several steps of DML-CZ workflow, as introduced and tested developed during the project development. We carried out most of the steps ourselves, to gain expertize and retain control of fine details, allowing us to plug-in new modules arising from leading edge research in the future—there are, currently, many new developments appearing and much research underway in the digitisation area. It is advisable for smaller project to outsource most of the workflow steps.

The most time consuming and costly step is metadata handling and editing, and image transformation and editing (if it cannot be automated). Bare scanning costs amount to less than 10% of the total page costs, and even less when pages can be physically cut before being used for batch scanning.

The complexity of the full digitisation workflow should not be underestimated, especially when digitising heterogeneous sources—continuous and flexible workflow adaptation is a must.

We believe that the methods, algorithms and tools developed do represent important step towards a European (EuDML) or even world-wide framework for a digital mathematics library, evolved, bottom up, from smaller "pilot" projects.

This research has been partially supported by the grant reg. no. 1ET200190513 of the Academy of Sciences of the Czech Republic, by MŠMT grants MSM0021622419 and 2C06009. The author thanks other DML-CZ colleagues for fruitful discussions that led to the design of the workflow described there and to the paper reviewers for improvement suggestions. Drawing of Figure 1 by Mirek Bartošek is acknowledged.

References

[1] Miroslav Bartošek, Petr Kovář, and Martin Šárfy. DML-CZ Metadata Editor: Content Creation System for Digital Libraries. In Sojka [21], pages 139–151.

[2] Pascal Chevalier. i2S DigiBook Mag, issue no. 2, July 2002. `http://ww.i2s-bookscanner.com/pdf/digibook_mag_no2.pdf`.

[3] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[4] DML-CZ. Digitization metadata editor. `http://sourceforge.net/projects/dme/`, 2009.

[5] Ted Dunning. Statistical identification of language. Technical Report MCCS 94-273, New Mexico State University, Computing Research Lab, 1994.

[6] Carole Goble. Curating Services and Workflows: the Good, the Bad and the Downright Ugly, 2008. Keynote presented at ECDL 2008, http://www.ecdl2008.org/keynotes/.

[7] Allyn Jackson. The Digital Mathematics Library. *Notices Am. Math. Soc.*, 50(4):918–923, 2003.

[8] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[9] István Marosi and László Tóth. OCR Voting Methods for Recognizing Low Contrast Printed Documents. In *Proceedings of Second International Conference on Document Image Analysis for Libraries (DIAL 2006)*, pages 108–115, April 2006.

[10] Tomáš Mudrák. Digitalizace matematických textů (in Czech, Digitisation of Mathematical Texts). Master's thesis, Masaryk University, Brno, Faculty of Informatics, April 2006. https://is.muni.cz/th/60738/fi_m/?lang=en.

[11] Radovan Panák. Digitalizácia matematických textov (in Czech, Digitisation of Mathematical Texts). Master's thesis, Masaryk University, Brno, Faculty of Informatics, April 2006. https://is.muni.cz/th/60587/fi_m/?lang=en.

[12] Tomáš Pulkrábek. Obrazové transformace při digitalizaci textů (in Czech, Image Transformation during Digitisation). Master's thesis, Faculty of Informatics, 2008. Bachelor's Thesis Masaryk University, Brno, Faculty of Informatics, https://is.muni.cz/th/139908/fi_b/?lang=en.

[13] Yves Rangoni, Faisal Shafait, and Thomas M. Breuel. OCR Based Thresholding. In *Proceedings of MVA 2009 IAPR Conference on Machine Vision Applications*, pages 3–18, May 2009.

[14] Radim Řehůřek and Petr Sojka. Automated Classification and Categorization of Mathematical Knowledge. In Serge Autexier, John Campbell, Julio Rubio, Volker Sorge, Masakazu Suzuki, and Freek Wiedijk, editors, *Intelligent Computer Mathematics—Proceedings of 7th International Conference on Mathematical Knowledge Management MKM 2008*, volume 5144 of *Lecture Notes in Computer Science LNCS/LNAI*, pages 543–557, Berlin, Heidelberg, July 2008. Springer-Verlag.

[15] Michal Růžička. Automated Processing of TEX-typeset Articles for a Digital Library. In Sojka [21], pages 167–176.

[16] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

[17] Steven J. Simske and Xiaofan Lin. Creating Digital Libraries: Content Generation and Re-Mastering. In *Proceedings of First International Workshop on Document Image Analysis for Libraries (DIAL 2004)*, page 13, January 2004.

[18] Ray Smith, Chris Newton, and Phil Cheatle. Adaptive Thresholding for OCR: A Significant Test. Technical Report HPL-1993-22, HP Laboratories Bristol, March 1993.

[19] Petr Sojka. Towards Digital Mathematical Library: Optical Character Recognition of Mathematical Texts. In Julius Štuller and Zdenka Linková, editors, *Inteligentní modely, algoritmy a nástroje pro vytváření semantického webu*, pages 110–113, Prague, 2006. Ústav informatiky AV ČR.

[20] Petr Sojka. Workflow in the digital mathematics library project: How mathematics is stored and retrieved. In J. Paralič, J. Dvorský, and M. Krátký, editors, *Proceedings of Znalosti 2006*, pages 243–247. VŠB–Technická univerzita Ostrava, 2006.

[21] Petr Sojka, editor. *Towards Digital Mathematics Library—Proceedings of DML 2008*, Birmingham, UK, July 2008. Masaryk University.

[22] Petr Sojka, Radovan Panák, and Tomáš Mudrák. Optical Character Recognition of Mathematical Texts in the DML-CZ Project. Technical report, Masaryk University, Brno, September 2006. presented at CMDE 2006 conference in Aveiro, Portugal.

[23] Petr Sojka and Radim Řehůřek. Classification of Multilingual Mathematical Papers in DML-CZ. In Petr Sojka and Aleš Horák, editors, *Proceedings of Recent Advances in Slavonic Natural Language Processing—RASLAN 2007*, pages 89–96, Karlova Studánka, Czech Republic, 2007. Masaryk University.

[24] Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, and Toshihiro Kanahori. INFTY — An integrated OCR system for mathematical documents. In C. Vanoirbeek, C. Roisin, and E. Munson, editors, *Proceedings of ACM Symposium on Document Engineering 2003*, pages 95–104, Grenoble, France, 2003. ACM.

Error Bound for Harmonic Balance Method Using Gröbner Base

Takashi Hisakado[1] and Masakazu Yagi[1]

[1] Department of Electrical Engineering, Kyoto University,
Kyotodaigakukatsura, Nishikyo, 615-8510, Kyoto, Japan
hisakado@kuee.kyoto-u.ac.jp

Abstract

This paper describes algebraic representations of error bounds for harmonic balance methods. Because the error bound of the harmonic balance method is high dimensional variety, the calculation of the error bound by numerical approaches is time consuming. We introduce an algebraic approach to the representation of the error bound. The error bound of the harmonic balance method is calculated by Gröbner base.

1 Introduction

Harmonic Balance (HB) method is one of the most popular methods for analyzing periodic solutions of nonlinear ordinary differential equations. The method gives approximated solutions by truncated Fourier series. In order to guarantee the solutions by the HB method, several methods with homotopy invariance theorem are reported[1, 2, 3]. These methods guarantee the approximated solutions by bounded regions, called error bounds, within which the exact solutions must exist. We introduce a method by symbolic manipulations which gives the error bound by a single algebraic equation[4].

2 Harmonic Balance Method

We explain the HB method with Duffing equation defined by

$$\frac{\mathrm{d}^2 u}{\mathrm{d}\tau^2} + \mu \frac{\mathrm{d}u}{\mathrm{d}\tau} + u^3 = E\cos\tau \tag{1}$$

for simplicity although the method can be applied to higher order differential equation and higher order polynomial nonlinearity[3, 4]. Eq.(1) can be rewritten as

$$\begin{aligned} G^{-1}(s;\mu)u(\tau) &= v(\tau) - N[u(\tau)], \\ & v(\tau) \equiv E\cos\tau, \quad N[u(\tau)] \equiv u^3, \quad s \equiv \mathrm{d}/\mathrm{d}\tau, \quad G^{-1}(s;\mu) \equiv s^2 + \mu s. \end{aligned} \tag{2}$$

Period 2π solution of Eq.(1) is given by Fourier series;

$$u(\tau) \equiv \sum_{k=0}^{\infty} \mathrm{Re}\left[x_k^* e^{\mathrm{j}k\tau}\right] = \sum_{k=0}^{\infty} \Re\left[(x_{k\mathrm{r}} + \mathrm{j}x_{k\mathrm{s}})e^{\mathrm{j}k\tau}\right], \tag{3}$$

where $x_k^* \in \mathbb{C}$, $x_{k\mathrm{r}}, x_{k\mathrm{s}} \in \mathbb{R}$, $\mathbb{C}$ is a set of complex numbers, $\mathbb{R}$ is a set of real numbers, $x_{0\mathrm{s}} = 0$ and $\Re[\cdot]$ denotes the real part. We define projection operators K_{L} and K_{H} by

$$u_{\mathrm{L}}(\tau) \equiv K_{\mathrm{L}} u(\tau) \equiv \sum_{k=0}^{n} \Re\left[x_k^* e^{\mathrm{j}k\tau}\right], \quad u_{\mathrm{H}}(\tau) \equiv K_{\mathrm{H}} u(\tau) \equiv \sum_{k=n+1}^{\infty} \Re\left[x_k^* e^{\mathrm{j}k\tau}\right], \tag{4}$$

where the u_{L} and u_{H} represent the truncated Fourier series and ignored higher harmonics, respectively. Applying the operator L_{H} to Eq.(2), we obtain an equation $\mathrm{FE}(u_{\mathrm{L}})$ which gives exact solutions of low frequency component u_{L} as

$$\mathrm{FE}(u_{\mathrm{L}}) \equiv G^{-1}(s;\mu)u_{\mathrm{L}} - \{v(\tau) - K_{\mathrm{L}}N[u_{\mathrm{L}} + u_{\mathrm{H}}]\} = 0. \tag{5}$$

Eq.(5), however, contains unknown high frequency component u_{H}. Ignoring the u_{H} in Eq.(5), we obtain the harmonic balance equation $\mathrm{FH}(u_{\mathrm{L}})$ defined by

$$\mathrm{FH}(u_{\mathrm{L}}) \equiv G^{-1}(s;\mu)u_{\mathrm{L}} - \{v(\tau) - K_{\mathrm{L}}N[u_{\mathrm{L}}]\} = 0. \tag{6}$$

Because the u_{L} is represented by $\boldsymbol{x} \equiv (x_{0\mathrm{r}}, x_{1\mathrm{r}}, x_{1\mathrm{s}}, \ldots, x_{n\mathrm{r}}, x_{n\mathrm{s}}) \in \mathbb{R}^{2n+1}$, Eq.(6) is an algebraic equation of $(x_{0\mathrm{r}}, x_{1\mathrm{r}}, x_{1\mathrm{s}}, \ldots, x_{n\mathrm{r}}, x_{n\mathrm{s}})$.

3 Derivation of Error Bound

3.1 Homotopy invariance

In order to find out the error bound of solutions by HB method, we use the following lemma of the homotopy invariance theorem[1, 2].

Lemma 1 *Let Ω be an open bounded set in $\mathbb{R}^m$ and let $\boldsymbol{f}, \boldsymbol{g}$:$\overline{\Omega} \to \mathbb{R}^m$ be two continuous maps where $\overline{\Omega}$ denotes the closure of the set Ω. If*

$$\|\boldsymbol{f}(\boldsymbol{z}) - \boldsymbol{g}(\boldsymbol{z})\|_2 < \|\boldsymbol{f}(\boldsymbol{z})\|_2 \quad \forall \boldsymbol{z} \in \partial\Omega, \tag{7}$$

where $\partial\Omega$ denotes the boundary of the set Ω, then $\deg(\boldsymbol{f}, \Omega) = \deg(\boldsymbol{g}, \Omega)$ where we denote by $\deg(\boldsymbol{f}, \Omega)$ the degree of $\boldsymbol{f}$ with respect to Ω.

That is, if there exists a bounded region Ω containing a single solution of Eq.(6) and on the boundary $\partial\Omega$

$$\|\mathrm{FT}(u_{\mathrm{L}}) - \mathrm{FH}(u_{\mathrm{L}})\|_2 < \|\mathrm{FH}(u_{\mathrm{L}})\|_2 \tag{8}$$

holds, then the solution of Eq.(5) also exists in Ω.

3.2 Estimation of u_{H}

In order to calculate the left hand side of Eq.(8) we estimate the high frequency component u_{H}. Let λ be a positive number satisfying

$$\lambda \geq \left\| \frac{\mathrm{d}N[u]}{\mathrm{d}u} \right\|_\infty. \tag{9}$$

Applying the operator K_{H} to Eq.(2), we obtain the relation;

$$u_{\mathrm{H}} = -K_{\mathrm{H}}G(s;\mu)N[u_{\mathrm{L}} + u_{\mathrm{H}}]. \tag{10}$$

If $\lambda H \equiv \lambda \sup_{k>n} |G(\mathrm{j}k;\mu)| < 1$, where $H \equiv \sup_{k>n} |G(\mathrm{j}k;\mu)|$, is satisfied, then there exists a unique u_{H} [3]. Using the mean value theorem and the contraction mapping, we obtain the following relations;

$$\|u_{\mathrm{H}}\|_i \leq \frac{\lambda H}{1 - \lambda H}\|u_{\mathrm{L}}\|_i \quad \text{for} \quad i = 1, 2. \tag{11}$$

The inequality estimates the higher frequency component u_{H} by the lower frequency component u_{L} with respect to l^1 and l^2 norms.

3.3 Equation of error bound

Using the mean value theorem and the estimation of the higher frequency components (11), we estimates the left hand side of Eq.(8) as

$$\begin{aligned} \|\mathrm{FT}(u_\mathrm{L}) - \mathrm{FH}(u_\mathrm{L})\|_2 &= \|K_\mathrm{L}N[u_\mathrm{L}] - K_\mathrm{L}N[u_\mathrm{L}+u_\mathrm{H}]\|_2 \\ &\leq \lambda\|u_\mathrm{H}\|_2 \leq \frac{\lambda^2 H}{1-\lambda H}\|u_\mathrm{L}\|_2. \end{aligned} \tag{12}$$

From Eq.(8) and Eq.(12) we define the error bound for the HB method by

$$\frac{\lambda^2 H}{1-\lambda H}\|u_\mathrm{L}\|_2 = \|\mathrm{FH}(u_\mathrm{L})\|_2. \tag{13}$$

3.4 Equation for determination of λ

The estimation of the higher frequency component (11) gives the following relation;

$$\|u\|_1 \leq \|u_\mathrm{L}\|_1 + \|u_\mathrm{H}\|_1 \leq \left(1 + \frac{\lambda H}{1-\lambda H}\right)\|u_\mathrm{L}\|_1. \tag{14}$$

Thus, if we determine the variable λ as

$$\lambda = 3\left(1 + \frac{\lambda H}{1-\lambda H}\right)^2 {\|u_\mathrm{L}\|_1}^2 \tag{15}$$

$$\geq 3{\|u\|_1}^2 = \left\|\frac{\mathrm{d}N[u]}{\mathrm{d}u}\right\|_1 \geq \left\|\frac{\mathrm{d}N[u]}{\mathrm{d}u}\right\|_\infty, \tag{16}$$

then λ satisfies Eq.(9).

4 Error Bound by Gröbner Base

If we eliminate λ from Eq.(13) using Eq.(15), we obtain the error bound. In order to eliminate λ by symbolic manipulation efficiently, we define the following variables,

$$\begin{aligned} &\alpha(\boldsymbol{x}) \equiv \|u_\mathrm{L}(\tau)\|_1 = \sum_{k=0}^{n}\sqrt{x_{k\mathrm{r}}^2 + x_{k\mathrm{s}}^2}, \quad \beta(\boldsymbol{x}) \equiv {\|u_\mathrm{L}(\tau)\|_2}^2 = \sum_{k=0}^{n}\left(x_{k\mathrm{r}}^2 + x_{k\mathrm{r}}^2\right), \\ &\gamma(\boldsymbol{x};\mu,E) \equiv {\|\mathrm{FH}(u_\mathrm{L})\|_2}^2 = \sum_{k=0}^{n}\left(f_{k\mathrm{r}}^2(\boldsymbol{x};\mu,E) + f_{k\mathrm{s}}^2(\boldsymbol{x};\mu,E)\right), \end{aligned} \tag{17}$$

where $f_{k\mathrm{r}}$ and $f_{k\mathrm{s}}$ denotes the real and imaginary part of kth Fourer components of $\mathrm{FH}(u_\mathrm{L})$, respectively. Next, we rewrite Eq.(15) and Eq.(13) by polynomial equation of $(\lambda,\alpha,\beta,\gamma)$ with the parameter H respectively as

$$f_{\mathrm{EB1}}(\lambda,\alpha;H) = \lambda(1-\lambda H)^2 - 3\alpha^2 = 0, \tag{18}$$

$$f_{\mathrm{EB2}}(\lambda,\beta,\gamma;H) = \lambda^4 H^2\beta - (1-\lambda H)^2\gamma = 0. \tag{19}$$

Now, using Gröbner base of order $\lambda \succ (\alpha,\beta,\gamma)$, we eliminate λ from Eqs.(18) and (19), and obtain the following error bound:

$$\begin{aligned} g_{\mathrm{EB}}(\alpha,\beta,\gamma;H) = {} & 9\alpha^4H^6\gamma^3 - 135\alpha^4\beta H^4\gamma^2 - 6\alpha^2\beta H^3\gamma^2 - 270\alpha^6\beta^2H^3\gamma + 225\alpha^4\beta^2H^2\gamma \\ & -30\alpha^2\beta^2H\gamma + \beta^2\gamma - 81\alpha^8\beta^3H^2 = 0. \end{aligned} \tag{20}$$

It is noted that Eq.(20) does not explicitly depend on n, $G(s;\mu)$ and parameters E and μ. Substituting Eq.(17) into Eq.(20), we obtain the error bound represented by $\boldsymbol{x}$.

5 Parameter Dependency of Error Bound

Because Eq.(20) symbolically contains the parameters E and μ, it is easy to estimate the parameter dependency of the error bound. In order to estimate the error bound which is a $2n$-dimensional variety in $\mathbb{R}^{2n+1}$, we use an approximation by quadratic form[4]. Figure 1 represents the parameter dependency of the error bounds with respect to E. We can observe that the error bounds are broken around the bifurcation points by the collision of two error bounds. The exact point of the collision is also calculated from Eq.(20) [4]. The lines E_{B1} and E_{B2} in Fig.1 show the exact break point of the error bounds.

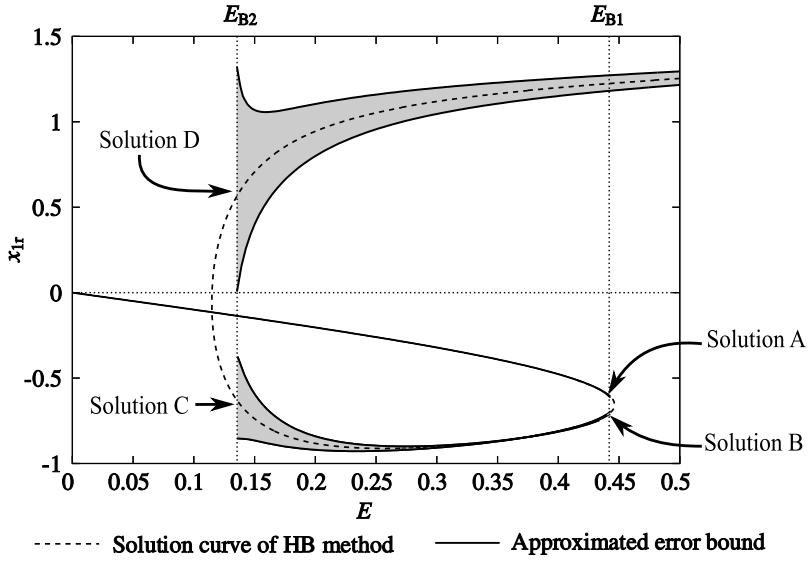


Figure 1: Parameter dependency of the approximated error bound.

References and Notes

[1] A. R. Bergen and R. L. Franks, "Justification of the describing function method," SIAM J. Contr. Optimiz., vol.9, pp.568-569, 1971.

[2] A. I. Mees and A. P. Bergen, "Describing functions revisited," IEEE Trans. Automat. Contr., vol.AC-20, pp.473-478, 1975.

[3] F. L. Swern, "Analysis of Oscillations in Systems with Polynomial-Type Nonlinearities Using Describing Functions," IEEE Trans. Automat. Contr., vol.AC-28, no.1, 1983.

[4] M. Yagi, T. Hisakado and K. Okumura: "An algebraic Approach to Guarantee Harmonic Balance Method Using Gröbner Base," IEICE Trans. Fundamentals, Vol.E91-A, No.9, pp.2442-2249, 2008.

Computer Assisted Proofs for Spectral Problems

Kaori Nagatou

Faculty of Mathematics, Kyushu University, 744, Motooka, Nishi-ku, Fukuoka, 819-0395, JAPAN / PRESTO Japan Science and Technology Agency
nagatou@math.kyushu-u.ac.jp

Abstract

Numerical methods which assure the reliability of the numerical results obtained by a computer, which sometimes are also called "Numerical Verification Methods", have attracted a great deal of attention in recent years. These methods assure not only a bound for the error between an approximate solution and an exact solution, but also prove the existence of an exact solution within the computed error bounds. This is why such methods are also called "Computer-Assisted Proofs". In particular, cases can be covered where purely analytical methods have failed. In this talk, we will show how guaranteed bounds for eigenvalues (together with eigenvectors) are obtained and how non-existence of eigenvalues in a concrete region could be assured. Some examples for several types of operators in bounded and unbounded domains will be presented.

1 Introduction

Up to now we have developed a method to enclose and exclude eigenvalues for differential operators [4, 5, 6, 7, 8, 9]. This method is based on Nakao's theory known as a numerical verification method for partial differential equations [10, 11, 12, 13], and it has a merit that it could be applied even in case the operator is not self-adjoint. A remarkable point of this eigenvalue enclosing/excluding is to assure an existence and non-existence range of eigenvalues with mathematically rigorous sense. This means not only a reliability of computed eigenpairs but also that such evaluation of eigenvalues (and eigenvectors) can be applied to related another problems, e.g. another numerical verification methods for nonlinear problems or stability analysis of bifurcation phenomenon in hydrodynamics.

This talk aims to show how eigenvalues (and eigenvectors) are enclosed or excluded in mathematically rigorous sense. In Sections 2 and 3 the principle of our eigenvalue enclosing/excluding method is presented, and in Section 4 we introduce some successful applications.

2 Eigenvalue enclosing method

Several methods to compute rigorous upper and lower bounds for eigenvalues of *symmetric* operators have been proposed so far, e.g. Krylov-Weinstein's bounds [2], Kato-Temple's bounds [3], Rayleigh-Ritz bounds [2], Lehman's Bounds [1], Homotopy Method [14] etc.

We have also developed a method to enclose eigenvalues and eigenvectors for differential operators [4, 5, 6], which was based on Nakao's verification methods for nonlinear differential equations [10, 11, 12, 13]. Our method is also applicable to non-symmetric operators. So far we have applied our enclosure method to enclose eigenpair of symmetric operators and to enclose real eigenvalues and corresponding eigenvectors of a non-symmetric operator. In the below we describe the principle of our method.

When we consider an eigenvalue problem $Lu = \lambda u$ for a linear operator L in bounded or unbounded domains, we treat it as a *nonlinear* system as follows:

$$\text{Find } (u, \lambda) \in V \times \mathbb{C} \text{ s.t.}$$

$$\begin{cases} Lu &= \lambda u, \\ \|u\| &= 1, \end{cases} \tag{1}$$

here V is a certain Sobolev space which is chosen depending on the property of the eigenfunction u. Then, by using a suitable operator F, we transform (1) into an equivalent fixed point equation

$$w = F(w) \quad \text{in } V \times \mathbb{C} \tag{2}$$

for $w = (u, \lambda)$. In order to treat the infinite dimensional equation (2) in a computer, we consider a finite dimensional subspace $V_h \subset V$ and a projection $P_h : V \to V_h$. Then we decompose (2) into the finite and the infinite dimensional parts:

$$\begin{cases} P_h w &= P_h F(w), \\ (I - P_h)w &= (I - P_h)F(w), \end{cases} \tag{3}$$

here I is the identity map on V. And we use the Newton-like method only for the former part, and the latter part is estimated by using an error estimation for P_h. If we construct a candidate set W suitable for Banach's fixed point theorem, then we obtain the local uniqueness and even the simplicity of the enclosed eigenvalue.

3 Eigenvalue excluding method

We have also proposed a method to *exclude* an eigenvalue in a concrete region, i.e. to prove that there is no eigenvalue in such an interval. This can be done as follows.

Let Λ be a small region in which we want to exclude any eigenvalues. Then consider a *linear* equation

$$Lu = \Lambda u. \tag{4}$$

Since the equation (4) has a trivial solution $u \equiv 0$, if we could prove the uniqueness of the solution of (4) then the non-existance of eigenvalues in Λ could be confirmed. In order to prove the uniqueness we construct an equivalent fixed point equation $u = T_\Lambda(u)$ as in the case of enclosing method. If there exists a non-empty, closed, bounded and convex set $U \subset V$ satisfying $\overline{T_\Lambda(U)} \subset \mathrm{int}(U)$, then there exists a unique solution $u \in V$ of $T_\Lambda u = u$. Here V is a certain Sobolev space. This assertion can be easily derived by the linearity of T_Λ. (See [4, 5] for details.)

4 Applications

Here we briefly introduce some successful applications. In [6] we proposed a numerical method to enclose eigenvalues and eigenfunctions of second-order elliptic operators, proving also local uniqueness properties. We numerically constructed a set containing *eigenpairs* which satisfied the hypotheses of Banach's fixed point theorem in a certain Sobolev space, by using a finite element approximation and constructive error estimates. We then proved the local uniqueness *separately* for eigenvalues and eigenfunctions. This local uniqueness assures the simplicity of the eigenvalue.

As an example in an unbounded domain, we studied the eigenvalue problem for an operator which defines a sort of non-trivial coupling of usual harmonic oscillators [8]. Since one has only a limited understanding of its eigenvalues, of the behavior of eigenfunctions, and in particular of the multiplicity of eigenvalues, here we tried to make a numerical approach to this system. More precisely, applying a numerical enclosure method for elliptic eigenvalue problems which is based on a verification procedure for nonlinear elliptic equations, adapted to such coupled-type eigenvalue problems on an unbounded domain, we developed a verified numerical computation for eigenvalues which also gave guaranteed information about multiplicity.

There exists a huge number of references concerned with bifurcations and stability results for the Navier-Stokes equations. Only a few, however, provide a rigorous result which guarantees stability or instability. Our aim in [9] was to present a rigorous theorem which proves the stability of certain solutions arising in what is called the Kolmogorov problem. We accomplish this by verified computation. The eigenvalue problem arising in the Kolmogorov problem is not self-adjoint and, accordingly, it is quite difficult to treat theoretically. Our method is a rigorous numerical approach to deal with this difficulty, and numerical examples were given as a demonstration.

As the most recent application we treat Eigenvalue excluding for 1-D Schrödinger operators. We consider a 1-D Schrödinger operator on the whole real line, with a potential which is a sum of a periodic function and some decaying perturbation. This kind of operator has essential spectrum with band-gap structure, and depending on the perturbation it may have isolated eigenvalues in the spectral gaps. Due to the lack of appropriate variational characterizations and to the "spectral pollution" problem, it is difficult to locate these eigenvalues analytically or numerically. Here we focus on excluding eigenvalues in spectral gaps and show how a mathematically rigorous treatment of such a problem can be done by numerical verification.

The details for these applications will be presented in the talk.

References and Notes

[1] Behnke, H., and Goerisch, F., Inclusions for Eigenvalues of Selfadjoint Problems, In: J.Herzberger(eds.), Topics in Validated Computations-Studies in Computational Mathematics, Elsevier, Amsterdam, 1994.

[2] Chatelin, F., Spectral Approximation of Linear Operators, Academic Press, New York, 1983.

[3] Kato, T., On the upper and lower bounds of eigenvalues, Journal of the Physical Society of Japan, 4 (1949), pp. 334-339.

[4] Nakao, M. T., Yamamoto, N., and Nagatou, K., Numerical Verifications for eigenvalues of second-order elliptic operators, Japan Journal of Industrial and Applied Mathematics, 16, No.3 (1999), pp. 307-320.

[5] Nagatou, K., Yamamoto, N., and Nakao, M. T., An approach to the numerical verification of solutions for nonlinear elliptic problems with local uniqueness, Numerical Functional Analysis and Optimization, 20, 5 & 6 (1999), pp. 543-565.

[6] Nagatou, K., A numerical method to verify the elliptic eigenvalue problems including a uniqueness property, Computing, 63 (1999), pp. 109-130.

[7] Nagatou, K., and Nakao, M. T., An enclosure method of eigenvalues for the elliptic operator linearlized at an exact solution of nonlinear problems, a special issue of Linear Algebra and its Applications on LINEAR ALGEBRA IN SELF-VALIDATING METHODS, 324/1-3 (2001), pp. 81-106.

[8] Nagatou, K., Nakao, M. T., and Wakayama, M., Verified numerical computations for eigenvalues of non-commutative harmonic oscillators, Numerical Functional Analysis and Optimization, 23, 5 & 6 (2002), pp. 633-650.

[9] Nagatou, K., A computer-assisted proof on the stability of the Kolmogorov flows of incompressible viscous fluid, Journal of Computational and Applied Mathematics, 169/1 (2004), pp. 33-44.

[10] Nakao, M.T., A numerical approach to the proof of existence of solutions for elliptic problems, Japan Journal of Applied Mathematics **5** (1988), pp. 313-332.

[11] Nakao, M.T., A numerical approach to the proof of existence of solutions for elliptic problems Ⅱ, Japan Journal of Applied Mathematics **7** (1990), pp. 477-488.

[12] Nakao, M.T. and Yamamoto, N., Self-validating methods (in Japanese), Nihonhyoron-sha, 1998.

[13] Nakao, M.T., Numerical verification methods for solutions of ordinary and partial differential equations, Numerical Functional Analysis and Optimization **22** (3&4) (2001), pp. 321-356.

[14] Plum, M., Eigenvalue inclusions for second-order ordinary differential operators by a numerical homotopy method, Journal of applied mathematics and physics (ZAMP), 41 (1990), pp. 205-226.

Rigorous numerics for homoclinic dynamics

Daniel Wilczak[1,2*]

[1] Department of Mathematics, University of Uppsala,
Box 480 Uppsala, Sweden

[2] Institute Of Computer Science, Jagiellonian University,
Łojasiewicza 6, 30-348 Kraków, Poland
wilczak@ii.uj.edu.pl

1 Introduction

Homoclinic and heteroclinic connections of hyperbolic objects play an important role in the study of dynamical systems from a global point of view. They were used in the design of space missions using libration point dynamics [1], among which the Genesis [2] has been the first one to make use of a heteroclinic connection.

For the design of such missions, the circular Restricted Three Body Problem (RTBP) is the natural problem to start with. Of special interest are the L_1 and L_2 libration points because of their suitability to place stationary satellites.

It turns out that the existence of homoclinic and heteroclinic orbits as well as chaotic dynamics for a given parameter values of the system is rather difficult to prove by means of an analytic approach. In this report we propose a method that allows us to verify the existence of homoclinic and heteroclinic orbits for maps and flows by means of a computer assisted proof. The method is geometric in the spirit and it assumes that the map is of class C^1 and that we can compute rigorous bounds for values and derivatives of the map. The method has been applied to the Planar Circular Restricted Three Body Problem (PCR3BP).

Let S and J be two bodies called Sun and Jupiter, of masses $m_s = 1-\mu$ and $m_j = \mu$, $\mu \in (0,1)$, respectively. They rotate in the plane on circles counter clockwise about their common center of mass and with the angular velocity normalized to one. Choose a rotating coordinate system, so that the origin is at the center of mass and the Sun and the Jupiter are fixed on the x-axis at $(-\mu, 0)$ and $(1-\mu, 0)$, respectively. In this coordinate frame the equations of motion of a massless particle called the comet or the spacecraft under the gravitational action of the Sun and the Jupiter are (see [3] and references given there)

$$\ddot{x} - 2\dot{y} = \Omega_x(x,y), \qquad \ddot{y} + 2\dot{x} = \Omega_y(x,y), \tag{1}$$

where

$$\Omega(x,y) = \frac{x^2+y^2}{2} + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} + \frac{\mu(1-\mu)}{2}$$
$$r_1 = \sqrt{(x+\mu)^2+y^2}, \qquad r_2 = \sqrt{(x-1+\mu)^2+y^2}.$$

Equations (1) are called the equations of the Planar Circular Restricted Three-Body Problem (PCR3BP). They have a first integral called the *Jacobi integral*, which is given by

$$C(x,y,\dot{x},\dot{y}) = -(\dot{x}^2+\dot{y}^2) + 2\Omega(x,y).$$

*Correspondence to: Department of Mathematics, University of Uppsala,
Box 480 Uppsala, Sweden, Phone: 018-471 3200, Fax: 018-471 3201.

We consider the PCR3BP on the hypersurface

$$\mathcal{M}(\mu, C) = \{(x, y, \dot{x}, \dot{y}) \,|\, C(x, y, \dot{x}, \dot{y}) = C\}$$

and we restrict our attention to the following parameter values $C = 3.03$, $\mu = 0.0009537$ - the parameter values for the *Oterma* comet in the Sun-Jupiter system (see [3]).

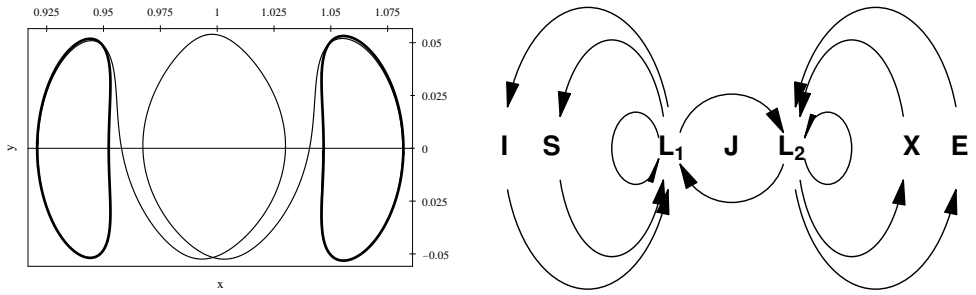


Figure 1: Left: the primary heteroclinic connection between L_1 and L_2 orbits. Right: the graph of symbolic dynamics for the PCR3BP.

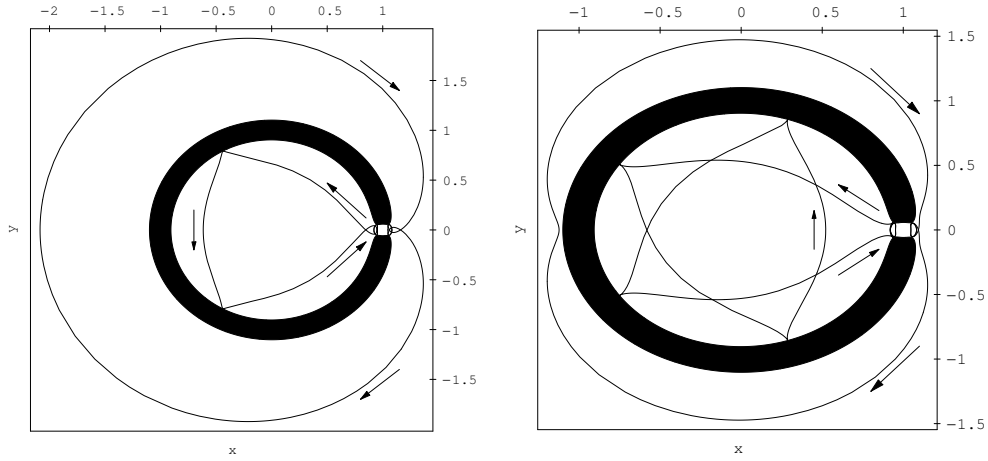


Figure 2: Homoclinic orbits to L_1 and L_2 orbits. The projection of the energy surface $\mathcal{M}(\mu, C)$ for $\mu = 0.0009537$ and $C = 3.03$ onto (x, y) coordinates does not contain the C-shaped black region. In this region a comet cannot move due to its energy level.

Theorem 1 *[5, 6] For the parameter values $C = 3.03$, $\mu = 0.0009537$ there are two periodic solutions of the equation (1) around the libration points L_1^* and L_2^* called the Lyapunov orbits and denoted by L_1 and L_2, respectively. Moreover, there exist*

- *heteroclinic orbits in both directions connecting L_1 and L_2 – see Fig. 1 left panel,*
- *two geometrically different homoclinic solutions in the Sun region to the L_1 orbit, denoted by S and I – see Fig. 2,*

- *two geometrically different homoclinic solutions in the exterior region to the L_2 orbit, denoted by X and E – see Fig. 2.*

Moreover, for any biinfinite path on the graph presented in Fig. 1 (right panel)

$$\mathbf{a} = (a_i)_{i\in\mathbb{Z}} \in \{L_1, L_2, X, E, I, S\}^{\mathbb{Z}}$$

- *there exists a true orbit of the PCR3BP which stays close (explicitly given estimation) to the orbits $\{L_1, L_2, X, E, I, S\}$ with the order given by the sequence $\mathbf{a}$,*
- *if the sequence $\mathbf{a}$ is periodic then the corresponding solution to PCR3BP can be chosen to be periodic,*
- *if the sequence $\mathbf{a}$ has the form*

$$(\ldots, L_k, L_k, a_0, a_1, \ldots, a_N, L_s, L_s, \ldots)$$

then there exists a solution $u(t)$ of the equation (1) such that

  - *the omega limit set $\omega(u) = L_s$,*
  - *the alpha limit set $\alpha(u) = L_k$,*
  - *there is a part of trajectory of u which stays close to the orbits $(a_0, a_1, \ldots, a_N)$ with the preserved order.*

2 Topological tools.

The main idea of the proof of Theorem 1 is to use the method of covering relations in order to prove the existence of orbits which intersect given subsets of the phase space with a desired order. The existence of homoclinic and heteroclinic orbits requires an argument for the convergence of an orbit. This will be verified by means of the cone conditions and presented in the next section.

Definition 1 *[7] Let N be a parallelogram with distinguished left and right halfplanes N^r and N^l and corresponding right and left edges N^{re}, and N^{le} as presented in Fig. 3 (left panel). We will call such an object an h-set.*

Let N, M be h-sets and let $f\colon N \to \mathbb{R}^2$ be continuous.

Definition 2 *[7] We say that N f-covers M and denote this by $N \overset{f}{\Longrightarrow} M$ if*

- $f(N) \subset \operatorname{int}(M \cup M^r \cup M^l)$
- *$f(N^{re})$ and $f(N^{le})$ are mapped into different halfplanes* $\operatorname{int}(M^r)$ *and* $\operatorname{int}(M^l)$.

The geometry of this concept is presented in Fig. 3 - right panel.

The following theorem is the main tool used in this paper for proving the existence of chaotic dynamics.

Theorem 2 *[7] Assume $\{N_i\}_{i=1,\ldots,K}$ are pairwise disjoint h-sets. Let $(i_j)_{\mathbb{Z}}$ be a biinfinite sequence such that*

$$N_{i_j} \overset{f}{\Longrightarrow} N_{i_{j+1}}.$$

Then there exists a sequence $(x_j)_{j\in\mathbb{Z}}$ such that $f(x_j) = x_{j+1}$ and $x_j \in \operatorname{int}(N_{i_j})$ for $j \in \mathbb{Z}$. If the sequence $(i_j)_{\mathbb{Z}}$ is periodic then x_0 can be chosen to be a periodic point for f with the same principal period.

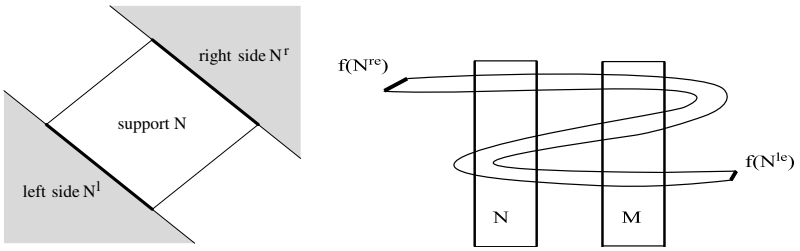


Figure 3: Left: an h-set. Right: geometry of the covering relations. In this case $N \overset{f}{\Longrightarrow} M$ and $N \overset{f}{\Longrightarrow} N$.

3 Cone conditions.

Let $Q\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ be a quadratic form. An h-set N with associated quadratic form Q_N will be called an h-set with cones and denoted by (N, Q_N).

Definition 3 *[4] Let $f\colon N \to \mathbb{R}^2$ be a smooth map. Let (N, Q_N) and (M, Q_M) be h-sets with cones. We will say that f satisfies the cone conditions with respect to the pair (N, M) if for $x, y \in N$ holds*

$$Q_M(f(x) - f(y)) > Q_N(x - y).$$

The following theorem can be used to prove the convergence of a trajectory to a fixed point.

Theorem 3 *[4] Let (N, Q_N) be an h-set with cones. Assume that $N \overset{f}{\Longrightarrow} N$ and f satisfies the cone conditions with respect to the pair (N, N). Then*

- *the map f has unique fixed point x_* in N,*
- *if $x \in N$ is such that $f^k(x) \in N$ for $k \in \mathbb{N}$ then $\lim_{k\to\infty} f^k(x) = x_*$,*
- *if $(x_k)_{k\leq 0}$ is such that $f(x_{k-1}) = x_k$ and $x_k \in N$ for $k \leq 0$ then $\lim_{k\to-\infty} x_k = x_*$.*

4 Proof of Theorem 1.

The proof of Theorem 1 consists of the following steps which were verified by means of verified numerics.

- After fixing the Jacobi constants C and the mass ratio μ corresponding to the Oterma comet the motion is on three dimensional isoenergetic manifold $\mathcal{M}(\mu, C)$. On this manifold we choose a Poincaré section $\Pi = \mathcal{M}(\mu, C) \cap \{y = 0\}$.
- We construct two h-sets with cones (H_1, Q_1), (H_2, Q_2) on Π and verified that Theorem 3 apply to a suitable Poincaré map P and the sets (H_i, Q_i), $i = 1, 2$. This proves the existence of two periodic orbits (called Lyapunov orbits) L_1 and L_2 which intersect the sets H_1 and H_2, respectively, at exactly one point.
- We construct six chains of h-sets along homoclinic and heteroclinic orbits presented in Fig 1 and Fig. 2 such that

$$H_{k_i} \overset{P}{\Longrightarrow} M_{i,1} \overset{P}{\Longrightarrow} \cdots \overset{P}{\Longrightarrow} M_{i,N_i} \overset{P}{\Longrightarrow} H_{s_i},$$

for $i = 1, \ldots, 6$, where P is a suitable Poincaré map. The cases $k_i \neq s_i$ correspond to heteroclinic connections between L_1 and L_2 in both directions. The cases $k_i = s_i = 1$ and $k_i = s_i = 2$ correspond to two pairs of homoclinic orbits in the interior and the exterior regions, respectively.

Now the assertion follows from Theorems 2 and 3.

We would like to point out here that the verification of the cone conditions required verified integration of the variational equations associated to the equations for the PCR3BP.

References and Notes

[1] D. W. Dunham and R. W. Farquhar, *Libration point missions*, in Libration Point Orbits and Applications, G. Gómez, M. W. Lo, and J. J. Masdemont, eds., 1978–2002, 2003.

[2] K. C. Howell, B. T. Barden, R. S. Wilson, and M. W. Lo, *Trajectory design using a dynamical systems approach with application to GENESIS*, Advances in the Astronautical Sciences 97, 1665–1684, (1998).

[3] W. S. Koon, M. W. Lo, J. E. Marsden and S. D. Ross, *Heteroclinic Connections between Periodic Orbits and Resonance Transitions in Celestial Mechanics*, Chaos 10, 427–469, (2000).

[4] H. Kokubu, D. Wilczak and P. Zgliczyński, *Rigorous verification of cocoon bifurcations in the Michelson system*, Nonlinearity 20, 2147–2174, (2007).

[5] D. Wilczak and P. Zgliczyński, *Heteroclinic Connections between Periodic Orbits in Planar Restricted Circular Three Body Problem - A Computer Assisted Proof*, Comm. Math. Phys. 234, 37–75, (2003).

[6] D. Wilczak and P. Zgliczyński, *Heteroclinic Connections between Periodic Orbits in Planar Restricted Circular Three Body Problem - Part II*, Comm. Math. Phys. 259, 561–576, (2005).

[7] P. Zgliczyński, *Computer assisted proof of chaos in the Rössler equations and the Hénon map*, Nonlinearity 10, 243–252, (1997).

Construction of an automatic validated computation for boundary value problems of ODEs

Nobito Yamamoto [1*], Ryuji Ukawa[1], and Nozomu Matsuda[1]

[1] The University of Electro-Communications,
1-5-1 Chofu-ga-oka, Chofu, Tokyo, Japan
yamamoto@im.uec.ac.jp

Abstract

Our aim is to construct a computer program for validated computation on boundary value problems (BVPs) of ODEs. The user gives an equation and boundary conditions, and set some parameters, then the program gives a validated results for the equation. It is not necessary for the user to write complicated codes in order to adjust the program to his equation.

For validated computation of BVPs, we use interval arithmetic with INTLAB, Taylor expansion method, multiple-Newton method, and Brouwer's fixed point theorem. Most part of the validated process is based on a paper of R.J.Lohner [1], but a quasi-Newton method is adopted instead of the interval Newton method in his paper. This change reduces the cpu time remarkablly, however needs complicated program codes to install. Symbolic Math Tool Box in Matlab can be used to write the program codes almost automatically.

Moreover we make some device to treat large size matrices which appaer in validated computation of the quasi-Newton method.

1 Introduction

R.J Lohner introduced a validated computation method for boundary value problems in [1]. The method is based on Taylor expansion with polynomials of some degree. But it is pointed out by himself that the method needs interval matrices for systems which should be solved by validated computation, and that this may cause some difficulty in practical computation, because the sizes of the systems depend on the number of time steps and are usually large. The cpu time for solving such large size interval systems will become so much.

We propose a method without using interval matrices. This method gives a better performance than Lohner's method but needs more complicated program codes. In order to reduce human effort, we make automatic program generators using a computer algebra system so called Symbolic Math Tool Box which can be invoked together with MATLAB and INTLAB.

The authors will discuss

- An application of the Lohner method using non-interval matrices in the systems and comparison with the original method
- Some device to treat large size matrices which appear in the systems
- Automatic formulation for installing programs using Symbolic Math Tool Box in MATLAB combined with INTLAB

All computation was carried out by R.Ukawa and N.Matsuda.

*Correspondence to: Nobito Yamamoto, 1-5-1 Chofu-ga-oka, Chofu, Tokyo, Japan, TEL:+81424435349

.

2 Problem

Consider a system of ODEs. Find $\boldsymbol{u}(t) \in \mathbb{R}^m$ such that

$$\begin{cases} \dfrac{d\boldsymbol{u}}{dt} &= \boldsymbol{f}(t, \boldsymbol{u}), \qquad a < t < b, \\ \mathbf{r}(\mathbf{u}(a), \mathbf{u}(b)) &= 0 \end{cases}$$

holds. Here,

$\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$

$\boldsymbol{r} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$

should be differentiable w.r.t. $\boldsymbol{u}$.

3 Validated Computation

Now we describe a method of validated computation for our problem.

Take mesh points by

$$a = t_0 < t_1 < \cdots < t_{m-1} < t_m = b, \qquad m \in \mathbf{N}.$$

Consider n dimensional vectors $\mathbf{s_0}, \mathbf{s_1}, \cdots, \mathbf{s_{m-1}}, \mathbf{s_m}$, and take a vector $\mathbf{s}$ of $n \times (m+1)$ dimension as

$$\mathbf{s} = (\mathbf{s_0}, \mathbf{s_1}, \cdots, \mathbf{s_{m-1}}, \mathbf{s_m})^T.$$

Solve the ODEs with an initial condition $\mathbf{u}(t_k) = \mathbf{s}_k$ at $t = t_k$, and let the solution at $t = t_{k+1}$ be denoted by

$$\mathbf{u}(t_{k+1}; t_k, \mathbf{s_k}).$$

In actual computation, we have to apply validated methods to get the solution $\mathbf{u}(t_{k+1}; t_k, \mathbf{s_k})$. The Lohner method is the most popular for this purpose. But in this case it is sufficient to use the Taylor expansion method without QR decomposition, since we can ignore Wrapping Effects for just 1 step computation.

Define $F(\mathbf{s})$ for $\mathbf{s}$ by

$$F(\mathbf{s}) := \begin{bmatrix} \mathbf{u}(t_1; t_0, \mathbf{s_0}) - \mathbf{s_1} \\ \mathbf{u}(t_2; t_1, \mathbf{s_1}) - \mathbf{s_2} \\ \vdots \\ \mathbf{u}(t_m; t_{m-1}, \mathbf{s_{m-1}}) - \mathbf{s_m} \\ r(\mathbf{s_0}, \mathbf{s_m}) \end{bmatrix}$$

We solve an equation

$$F(\mathbf{s}) \quad = \quad \mathbf{0}.$$

If the solution $\mathbf{s}$ exists, then the solution $\boldsymbol{u}(t)$ to the boundary problem of the ODEs also exists. Moreover $\mathbf{s_j}$ gives the values of $\boldsymbol{u}(t_j)$.

The equation $F(\mathbf{s}) = \mathbf{0}$ is transformed into Newton-type equations

$$\begin{aligned} \mathbf{s} &= N(\mathbf{s}) \\ &= \mathbf{s} - (F'(\tilde{\boldsymbol{s}}))^{-1}F(\mathbf{s}), \end{aligned}$$

where $\tilde{\boldsymbol{s}} \approx \mathbf{s}$.

For validated computation, we will find some interval vector $[\mathbf{s}]$ such that

$$N([\mathbf{s}]) \subset [\mathbf{s}]$$

holds, and use Brouwer's fixed point theorem.

Lohner's original equation gives the operator N by

$$N_L([\mathbf{s}]; \tilde{\boldsymbol{s}}) := \tilde{\boldsymbol{s}} - (F'([\mathbf{s}]))^{-1}F(\tilde{\boldsymbol{s}})$$

where $\tilde{\boldsymbol{s}}$ is the center of $[\mathbf{s}]$. Note that the system matrix is an interval matrix.

Instead we use Quasi-Newton equation.

$$N([\mathbf{s}]; \tilde{\boldsymbol{s}}) := (F'(\tilde{\boldsymbol{s}}))^{-1}(F'(\tilde{\boldsymbol{s}})[\mathbf{s}] - F([\mathbf{s}]))$$

where $\tilde{\boldsymbol{s}}$ is an approximate solution to $\mathbf{s}$. Note that the system matrix is not interval any longer.

- Lohner's original equation has an interval matrix, which need much cpu time in computation.
- Quasi-Newton equation has an advantage in cpu time.
- On the other hand, the implementation of a computer program for Quasi-Newton equation has some defect. See the Section 5.

4 Treatment of large size systems

In practical computation, the number of the steps m might become so large, 10,000 to 20,000 as usual. Then the system matrices in the Newton operators have large sizes. Indeed they are sparse, but the inverse matrices are full which we have to hold in the memory of our computers in order to solve the systems with validated computation. Usually this is impossible.

In our case, the Jacobian F' has a special form, almost block diagonal, then we can use a kind of sweep out method. But naive use of sweep out causes Wrapping Effect. In order to avoid Wrapping Effect, we use the mathod so called Top Down Algorithm in which the operations of matrix products are done before matrix-vector products.

5 Symbolic manipulation

Before the implementation of calculation of interval polynomials, we have to combine the linear parts to get rid of unnecessary width expansion of intervals ('Dependency Problem'). For complicated calculation of intervals, the mean value formula is the most popular countermeasure to dependency problem:

$$[g([\boldsymbol{v}])] \subset g(\hat{\boldsymbol{v}}) + [g'([\boldsymbol{v}])]([\boldsymbol{v}] - \hat{\boldsymbol{v}}),$$

where $\hat{\boldsymbol{v}} \in [\boldsymbol{v}]$.

On the Quasi-Newton equation, we have to apply the mean value formula to the term $F'(\tilde{\boldsymbol{s}})[\mathbf{s}] - F([\mathbf{s}])$. But this is not a light work because we have to treat Taylor expansion method w.r.t $[\mathbf{s}]$. Then we use a symbolic manipulator in MATLAB.

- Symbolic Math Toolbox is a package in MATLAB to treat symbolic transformation in computers.
- In order to substitute some numerical values for symbols, Symbolic Math Toolbox usually uses a command `subs`. But this command cannot be used for interval values in Intlab.
- We adopt some devices in order to substitute interval values to symbols.

Details of these devices will be shown in the talk.

References and Notes

[1] R.J.Lohner, *Enclosing the solutions of ordinary initial and boundary value problems*, In: Kaucher, C.U.E., Kulisch, U.(Eds.), Computer Arithmetic: Scientific Computation and Programming Languages, Teubner, Stuttgart, 1987

Simplification of the lattice based attack of Boneh and Durfee for RSA cryptoanalysis

Yoshinori Aono *

Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan, aono5@is.titech.ac.jp

Abstract

In this paper we present a new formulation and its simpler analysis of the lattice based attack of Boneh and Durfee for the RSA cryptography [2]. We follow the same approach of Boneh and Durfee, however we propose a new way of defining a lattice with which we can achieve the same solvable key bound $d < N^{0.292}$. Our lattice is represented as a lower triangle matrix, which makes its analysis much simpler that of [2]. We think that this simpler analysis would be useful for considering applications/generalisations of this approach. In fact, as an example of such applications, we give a way of attacking RSA secret key with a certain repetitive structure. We omitted the proof of some lemmas in this paper, see the full version [1] for the complete proof.

1 Introduction

In [2], Boneh and Durfee proposed a polynomial time attack by which we can recover the RSA secret key d from the public information (e, N) when $d < N^{0.292}$; in the following, we call the bound $d < N^{0.292}$ the *solvable key bound* of Boneh and Durfee or simply the *Boneh-Durfee bound*. The basic idea of the attack is based on the Coppersmith technique by which we can obtain small solutions of a modular equation such as $f(x_1, x_2, \ldots, x_n) \equiv 0 \pmod{W}$. The technique converts the problem of finding a small solution of the equation to the problem of solving a system of polynomial equations by a lattice reduction algorithm such as the LLL algorithm [10].

Here is more detail explanation of their approach. The goal is to obtain a small solution (x_0, y_0) of the following target equation to recover the secret key.

$$f_{\mathrm{BD}}(x, y) = -1 + x(y + A) \equiv 1 \pmod{e} \tag{1}$$

Here $A = N + 1$. From this first the following bivariate polynomials are defined

$$g_{i,j}(x, y) = \begin{cases} x^{i-j}(f_{\mathrm{BD}}(x, y))^i e^{m-i} & \text{for } i \geq j \\ y^{j-i}(f_{\mathrm{BD}}(x, y))^j e^{m-j} & \text{for } i < j \end{cases} \tag{2}$$

for a certain range of (i, j) and an integer m. These polynomials are converted to a lattice represented by a row echelon matrix L_{BD} defined by using the coefficients of $g_{i,j}(x, y)$ with some parameters. Then by using a lattice reduction algorithm, we obtain a system of polynomial equations from which we can compute polynomial number of candidates of the solution (x_0, y_0) numerically.

In this approach a technically crucial point is to design a matrix for a lattice with a small determinant. They showed that their matrix has a sufficiently small determinant; however, its analysis is complicated since the technique of geometrically progressive matrices, and it seems hard to apply for the other situations. The purpose of this paper is to give a new

*The author and this research was supported in part by JSPS Global COE program "Computationism as a Foundation for the Sciences".

way to construct a lattice with asymptotically the same determinant that is much simpler to analyse.

Since Boneh and Durfee's work, variants of their technique have been proposed. Blömer and May [3] proposed a new lattice based algorithm for attacking RSA with a short secret key. They constructed a lower triangle lattice by eliminating some columns from the original lattice; this makes simplify the determinant analysis. In [9], Jochemsz and May gave an algorithm for finding small roots of a multivariate modular/integer equation based on a generalised lattice construction strategy. Note that both algorithms, achieve a slightly weaker solvable key bound than the Boneh-Durfee bound.

In this paper we follow the strategy of Boneh and Durfee to give a new variation of the lattice based attack with a simpler analysis. We propose a conversion from the polynomials (2) to three-variable polynomials $G_{i,j}(x, y, z)$ when we construct lattice; on the other hand, Boneh and Durfee directly constructed the lattice from $g_{i,j}(x, y)$. Since we obtain a lower triangle matrix representation of our lattice, we can easily compute its determinant. Therefore, we give a new simple algorithm to achieve the Boneh-Durfee bound.

As the application of our analysis technique we consider the situation where an RSA secret key has a repetitive structure; more precisely, the situation that the secret key d (in its binary representation) is the repeat of r short bit string d_0. In this situation, we showed that we can recover the secret key when $\beta < (r + 3 - \sqrt{r^2 + 6r + 1})/4$ where $\beta = \log_N d_0$. For $r = 1$, we can see this is equivalent to the attack of Boneh and Durfee.

This paper is organised as follows: In section 2, we give basic symbols, notations, lemmas. We give our formulation of the lattice based attack in section 3 and its detailed analysis is explained in 4. In section 5, we demonstrate that our approach can be used to analyse the situation when a secret key has a repetitive structure.

2 Preliminaries

In this section, for the following discussions, we introduce some notations, state some known facts, and key technical lemmas.

We use standard RSA notations throughout this paper. A given RSA instance is defined by p, q, e, and d, where p and q are large primes, e is a public key, and d is the corresponding secret key. Let $N = p \times q$, and let $\varphi(N)$ be the Euler's function; here we may simply assume that $\varphi(N) = (p-1)(q-1)$. We assume that $\gcd(e, \varphi(N)) = 1$. The *key relation* between e and d is $ed \equiv 1 \pmod{\varphi(N)}$ from which we derive our target equation (1) by following the argument in [2].

The basic strategy of the lattice based attack is to convert the problem of recovering RSA to the problem of finding small solution of a modular equation; more precisely, this problem is to find a solution within a certain range of a modular equation such as $f(x, y) \equiv 0 \pmod{W}$ for a polynomial $f(x, y)$ and a nonnegative integer W. In general, solving modular equation is not easy, whereas there are some cases where we may be able to use the standard numerical method for solving this problem. The Howgrave-Graham lemma [8] provides us with one of such cases.

To state the Howgrave-Graham lemma, we introduce the following norm for bivariate polynomials and integers.

Definition 1. XY**-norm** *Let* $f(x, y) = \sum_{i,j} a_{i,j} x^i y^j$ *be a polynomial with integral coef-*

ficients, X and Y be natural numbers. We define the XY-norm of $f(x,y)$ by

$$||f(x,y)||_{XY} \stackrel{\text{def}}{=} \sqrt{\sum_{i,j} a_{i,j}^2 X^{2i} Y^{2j}}.$$

Lemma 1. (Howgrave-Graham [8]) *For any positive integers X, Y and W, let $f(x,y)$ be a bivariate polynomial consisting with w terms with integral coefficients such that the following holds*

$$||f(x,y)||_{XY} < \frac{W}{\sqrt{w}}.$$

Then we have

$$f(x,y) \equiv 0 \ (mod\ W) \Leftrightarrow f(x,y) = 0$$

within the range of $|x| < X$ and $|y| < Y$.

Note that $f(x,y) = 0$ clearly implies $f(x,y) \equiv 0 \pmod{W}$. What is important is its converse. This lemma guarantees that we can find all solutions of the modular equation within the range from the integral solutions of $f(x,y) = 0$ (if they exist).

Now we introduce some definitions and some lemmas about the lattice; we need to obtain a polynomial with a small XY-norm to use Lemma 1, and this problem can be reduced to a problem of finding a short vector in a lattice. Consider linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$, then the lattice with basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is defined by

$$L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} a_i \mathbf{b}_i \,\middle|\, a_i \in \mathbb{Z} \text{ for } i = 1, \ldots, n \right\}. \tag{3}$$

That is, the lattice is the set of integral linear combinations of its basis vectors.

The shortest vector problem is to find a vector $\mathbf{v} \in L(\mathbf{b}_1, \ldots, \mathbf{b}_n) \setminus \{\mathbf{0}\}$ such that $|\mathbf{v}| \leq |\mathbf{v}'|$ for $^{\forall}\mathbf{v}' \in L(\mathbf{b}_1, \ldots, \mathbf{b}_n) \setminus \{\mathbf{0}\}$. In other word, this problem is to find a non-zero vector having the minimum length in $L(\mathbf{b}_1, \ldots, \mathbf{b}_n)$. We know that a lattice basis reduction algorithm finds a good approximation of the problem by computing a reduced basis. We use the LLL algorithm [10], the most widely used lattice reduction algorithm, in our analysis. The two short vectors in the reduced basis described in the following theorem are important.

Theorem 1. *[2, Fact 3.3] Let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be a given linearly independent basis. Then the LLL algorithm can find linearly independent lattice vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ such that*

$$|\mathbf{v}_1| \leq 2^{(n-1)/4} |\det(L)|^{1/n} \text{ and } |\mathbf{v}_2| \leq 2^{n/2} |\det(L)|^{1/(n-1)}. \tag{4}$$

Here, $\det(L)$ is the determinant of the lattice which is defined by the determinant of a matrix representation of the lattice; this is actually defined by

$$\det(L) = \det \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{pmatrix}.$$

When we obtain a lower triangle matrix representation of a lattice, we can easily compute $|\det(L)|$ by the product of its diagonal elements. we convert the short vectors in the reduced basis to polynomials satisfying the sufficient condition of the Howgrave-Graham lemma.

We introduce a mapping for converting polynomials to vectors; since a lattice reduction algorithm is designed for vectors, while our targets are polynomials. We divide this mapping into two steps, named a vectorisation and an instantiation respectively. We introduce a way to map three-variable polynomials to vectors since we will consider three-variable polynomials in our construction. i

Definition 2. Polynomials ⇒ vectors

Let $\mathbf{K}$ *be a finite sequence of distinct three-variable monomials. We assume a linear order on this, and let it be fixed; for any* t*, let* $x^{i_t}y^{j_t}z^{k_t}$ *be the* t*-th monomial in this order. Then for any* $f(x,y,z) = \sum_{1\le t\le|\mathbf{K}|} a_t x^{i_t}y^{j_t}z^{k_t}$*, we map it to the following vector* $\mathbf{b}$*, which is called the* vectorisation *of* $f(x,y,z)$ *and is denoted as* $\mathcal{V}_{\mathbf{K}}(f)$*.*

$$\begin{array}{lcccccccccc} f(x,y,z) & = & a_1x^{i_1}y^{j_1}z^{k_1} & + & a_2x^{i_2}y^{j_2}z^{k_2} & + & \cdots & + & a_{|\mathbf{K}|}x^{i_{|\mathbf{K}|}}y^{j_{|\mathbf{K}|}}z^{k_{|\mathbf{K}|}} \\ & & \downarrow & & \downarrow & & & & \downarrow & & \\ \mathbf{b} & = (& a_1x^{i_1}y^{j_1}z^{k_1} & , & a_2x^{i_2}y^{j_2}z^{k_2} & , & \ldots & , & a_{|\mathbf{K}|}x^{i_{|\mathbf{K}|}}y^{j_{|\mathbf{K}|}}z^{k_{|\mathbf{K}|}} &). & \end{array}$$

We introduce a conversion named an instantiation and its inverse; it converts a three-variable monomials to integers by substitution. Our matrix will be defined by using the vectorizations and hence each element of the matrix is monomial. On the other hand, a lattice reduction algorithm is designed for integer lattices or integer matrices. Thus, for using a lattice reduction algorithm, we need to instantiate our matrix by substituting some integers X, Y and Z to x, y and z, which we call an *instantiation* with X, Y and Z. Conversely, converting an integer vector to a polynomial is called a *deinstantiation*. Note that (since $\mathbf{K}$ and the order of monomials is fixed) we know a monomial $x^{i_t}y^{j_t}z^{k_t}$ corresponding to the t-th entry of a given vector; hence, deinstantiation at the t-th entry can be achieved by simply dividing its integral value by $X^{i_t}Y^{j_t}Z^{k_t}$.

These vectorization, instantiation, and deinstantiation procedures are essentially the same as those used by Boneh and Durfee (except that we consider three-variable polynomials while bivariate polynomials have been used by them).

3 A New Lattice Based Algorithm

In this section we give a new lattice based algorithm for RSA with a short secret key; that is, a new lattice construction and its simpler analysis to derive the same Boneh-Durfee bound $d < N^{0.292}$. Our analysis requires only elementary lemmas and calculations. The detailed analysis is given in the next section. What is different from the original algorithm is to use three-variable polynomials to construct a lattice; the bivariate polynomials (2) are used directly in the original paper. We first state some definitions and lemma to explain our lattice construction.

Canonical Replacement

We convert the bivariate polynomials $g_{i,j}(x,y)$ to three-variable polynomials artificially. We first express $g_{i,j}(x,y)$ as a sum of monomials and then replace every xy by $z+1$ in $g_{i,j}(x,y)$. For example, the polynomial $g_{2,3}(x,y) = e^{m-2}(-1+xy+Ax)^2y = e^{m-2}y + A^2e^{m-2}x^2y - 2Ae^{m-2}xy - 2e^{m-2}xy^2 + 2Ae^{m-2}x^2y^2 + x^2y^3$ is converted to the polynomial $G_{3,2}(x,y,z) = e^{m-2}y + A^2e^{m-2}x(1+z) - 2Ae^{m-2}(1+z) - 2e^{m-2}y(1+z) + 2Ae^{m-2}(1+z)^2 + e^{m-2}y(1+z)^2$. Like this example, we will denote the converted polynomial from $g_{i,j}(x,y)$ by $G_{i,j}(x,y,z)$, and we call this conversion a *canonical replacement*. It is clear that $G_{i,j}(x,y,-1+xy) = g_{i,j}(x,y)$. Though artificial, this canonical replacement allows us to define a lower triangle matrix representation of our lattice.

Step 1:	Choose attack parameters m and δ.
Step 2:	Define an index sequence $\mathbf{I}$ and a monomial sequence $\mathbf{K}$ (as explained in Section 4). For each $(i,j) \in \mathbf{I}$, define a polynomial $g_{i,j}(x,y)$ as (2) and a polynomial $G_{i,j}(x,y,z)$ by the canonical replacement of $g_{i,j}(x,y)$. Construct a lattice L using vectors $\mathcal{V}_{\mathbf{K}}(G_{i,j})$ as row vectors in the order of (i,j) following $\mathbf{I}$.
Step 3:	Instantiate L with $X = \lfloor N^{\delta} \rfloor$, $Y = \lfloor N^{0.5} \rfloor$ (and $Z = \sqrt{X^2Y^2+1}$). Then apply a lattice reduction algorithm to it.
Step 4:	For two short vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ computed by a reduction algorithm, compute their deinstantiations $\mathbf{v}'_1$ and $\mathbf{v}'_2$. Define polynomials $H_1(x,y,z)$ and $H_2(x,y,z)$ by summing up the monomials in $\mathbf{v}'_1$ and $\mathbf{v}'_2$ respectively. Then define $h_1(x,y) = H_1(x,y,-1+xy)$ and $h_2(x,y) = H_2(x,y,-1+xy)$.
Step 5:	Enumerate all integral solutions of $h_1(x,y) = h_2(x,y) = 0$. For each of those solutions, compute d by (1) and check whether it is an integer.

Figure 1: Our version of the lattice based attack

Here we extend the notion of XY-norm for converted three-variable polynomials and show some useful bound. Though we consider such converted three-variable polynomials, they are essentially bivariate polynomials; hence, we still discuss its XY-norm and show an inequality. Let $F(x,y,z) = \sum_{i,j,k} b_{i,j,k} x^i y^j z^k$ be a three-variable polynomial. For this F, we define a three-variable version of the XY-norm as follows:

$$||F(x,y,z)||_{XY} \stackrel{\text{def}}{=} \sqrt{\sum_{i,j,k} b_{i,j,k}^2 X^{2i} Y^{2j} (X^2Y^2+1)^k}.$$

Again this definition is somewhat artificial; one motivation is to have the following bound.

Lemma 2. *Let $f(x,y)$ be any bivariate polynomial and let $F(x,y,z)$ be is a polynomial obtained by applying the canonical replacement to $f(x,y)$. Let v be the maximum degree of z in $F(x,y,z)$. Then for any non-negative integers X and Y, the following holds.*

$$||f(x,y)||_{XY} \leq (v+1)||F(x,y,z)||_{XY}.$$

Remark. We will assume that $Z = \sqrt{X^2Y^2+1}$ whenever we consider instantiation/deinstantiation with some X and Y to keep its consistent with this extended XY-norm notion. Thus, for any three-variable polynomial $F(x,y,z)$ obtained as a sum of monomials of the deinstantiation of some vector $\mathbf{F}$ w.r.t. X and Y, the following relation is immediate.

$$||F(x,y,z)||_{XY} = |\mathbf{F}|. \tag{5}$$

Now we explain our version of the lattice based attack for RSA following its outline stated in Figure 1. This is essentially the same as the one by Boneh and Durfee except for polynomials and a lattice construction.

We first define symbols used in the algorithm. Let δ be the ratio of the bit-length of d to that of N; here we assume that $\delta < 0.5$. Let m be an another parameter that is set as an integer greater than one; the larger m would yield the better solvable key range but the more computation time is necessary. The Boneh-Durfee bound $\delta < 0.292$ is the approximated value when we take sufficiently large m. Thus, considering available computational resource and δ, an appropriate number should be chosen for m.

Then we define the set $I = \{(i,j) \in \mathbb{Z}^2 | 0 \le i \le m, 0 \le j \le 2(1-\delta)i\}$. The sequence $\mathbf{I}$ is defined by introducing some order to elements in I; however we postpone its explanation to the next section. For $(i,j) \in I$, polynomials $g_{i,j}(x,y)$ are defined as follows:

$$g_{i,j}(x,y) \stackrel{\text{def}}{=} \begin{cases} x^{i-j}(f_{\mathrm{BD}}(x,y))^i e^{m-i} & \text{for } i \ge j \\ y^{j-i}(f_{\mathrm{BD}}(x,y))^j e^{m-j} & \text{for } i < j. \end{cases} \tag{6}$$

These are the same polynomials defined in [2]; more precisely, our $g_{i,j}(x,y)$ for $i \ge j$ and for $i < j$ correspond to their $g_{i,j}(x,y)$ and $h_{i,j}(x,y)$ respectively. We then further extend them to three-variable polynomials $G_{i,j}(x,y,z)$ by the canonical replacement. Consider the set of monomials of type $x^i y^j z^k$ that appear in some $G_{i,j}(x,y,z)$. Again its ordered version $\mathbf{K}$ will be defined in the next section. Now we let $\mathbf{b}_{i,j} = \mathcal{V}_{\mathbf{K}}(G_{i,j})$ and define our lattice L as follows:

$$L = \begin{bmatrix} \mathbf{b}_{0,0} \\ \vdots \\ \mathbf{b}_{m,m'} \end{bmatrix} \tag{7}$$

where m' is $\lfloor 2(1-\delta)m \rfloor$. One important point here is that we can choose some appropriate ordering for $\mathbf{K}$ so that L becomes lower triangle; we prove this in the next section.

Next we carry out a lattice reduction algorithm on L' that is obtained as the instantiation of L with parameters $X = \lfloor N^{\delta} \rfloor$ and $Y = \lfloor 3N^{0.5} \rfloor$ (and $Z = \sqrt{1 + X^2 Y^2}$); for our analysis, we consider the LLL algorithm. The algorithm computes a reduced basis, which contains short vectors in the lattice.

From two short vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ in the reduced basis, we construct the corresponding polynomials $h_1(x,y)$ and $h_2(x,y)$. First we convert the two vectors to their deinstantiations $\mathbf{v}'_1$ and $\mathbf{v}'_2$. Then define $H_1(x,y,z)$ and $H_2(x,y,z)$ as the sums of all monomials in $\mathbf{v}'_1$ and $\mathbf{v}'_2$ respectively. From the construction of L and the nature of the instantiation/deinstantiation, that is, we can easily see each $\mathbf{v}_c$ is a integral linear combination of $\mathbf{b}_i$'s; hence this yields that $H_1(x,y,z)$ and $H_2(x,y,z)$ are integral linear combinations of $G_{i,j}(x,y,z)$. We then obtain $h_1(x,y)$ and $h_2(x,y)$ by $h_c(x,y) = H_c(x,y,-1+xy)$ for $c = 1, 2$.

Finally, in step 5, we solve the simultaneous equation $h_1(x,y) = h_2(x,y) = 0$. For each (x_1, y_1) of the integral solutions of the equation, compute d by $d = (-1 + x_1(y_1 + A))/e$ and check it is indeed the correct secret key, i.e., check whether it is a non-negative integer. This is the outline of our version of the lattice based attack.

Now we show relationships between the polynomials computed in the algorithm; our target is to derive the Boneh-Durfee bound via considering a sufficient condition of the Howgrave-Graham lemma. By construction we have

$$f_{\mathrm{BD}}(x,y) \equiv 0 \pmod{e} \Rightarrow [g_{i,j}(x,y) = G_{i,j}(x,y,-1+xy) \equiv 0 \pmod{e^m} \text{ for } {}^{\forall}(i,j) \in I].$$

On the other hand, $[G_{i,j}(x,y,-1+xy) \equiv 0 \pmod{e^m} \text{ for } {}^{\forall}(i,j) \in I] \Rightarrow [H_c(x,y,-1+xy) = h_c(x,y) \equiv 0 \pmod{e^m} \text{ for } c = 1,2]$ since each $H_c(x,y,z)$ is an integral linear combination of $G_{i,j}(x,y,z)$. Thus, if both $h_1(x,y)$ and $h_2(x,y)$ satisfy the condition of the Howgrave-Graham lemma for X, Y and $W = e^m$, we have

$$h_c(x,y) \equiv 0 \pmod{e^m} \Leftrightarrow h_c(x,y) = 0 \text{ for } c = 1,2,\ |x| < X \text{ and } |y| < Y$$

and this implies

$$f_{\mathrm{BD}}(x,y) \equiv 0 \pmod{e} \Rightarrow h_c(x,y) = 0 \text{ for } c = 1,2,\ |x| < X \text{ and } |y| < Y. \tag{8}$$

Therefore a small solution of $f_{\mathrm{BD}}(x,y) \equiv 0 \pmod{e}$ must be included in the set of small solutions of $h_1(x,y) = h_2(x,y) = 0$ when h_1 and h_2 satisfy the Howgrave-Graham condition; more precisely, if for each $c = 1, 2$, h_c satisfies the following.

$$||h_c(x,y)||_{XY} < \frac{e^m}{\sqrt{w}} \tag{9}$$

Here w is the number of terms in $h_c(x,y)$, which is equal or less than $(1-\delta)m^2$.

Now we consider this condition for each $h_c(x,y)$ to derive the Boneh-Durfee bound. We have by Lemma 2,

$$||h_c(x,y)||_{XY} \leq (v+1)||H_c(x,y,z)||_{XY} \leq (m+1)||H_c(x,y,z)||_{XY}$$

where v is the maximum degree of z in $H_c(x,y,z)$, which is equal to or smaller than m. Moreover by Theorem 1 and (5) we have

$$||H_c(x,y,z)||_{XY} = |\mathbf{v}_c| \leq 2^{n/2} \det(L')^{1/(n-1)}.$$

Rearranging these conditions, we have a sufficient condition for the Howgrave-Graham lemma as follows.

$$(m+1)2^{n/2} \det(L')^{1/(n-1)} < \frac{e^m}{\sqrt{1-\delta}m}.$$

Following the analysis of Boneh and Durfee, we disregard the numbers $(m+1)2^{n/2}$ and $\sqrt{1-\delta}m$ because these are sufficiently small comparing to the RSA parameters, and we use $\det(L')^{1/n}$ instead of $\det(L')^{1/(n-1)}$. Hence we have our simplified sufficient condition.

$$\det(L')^{1/n} < e^m. \tag{10}$$

Here from the analysis of the next section, we have both

$$n = (1-\delta)m^2 + o(m^2) \text{ and } \det(L') = N^{\left(-\frac{1}{3}\delta^2 - \frac{1}{3}\delta + \frac{5}{6}\right)m^3 + o(m^3)}.$$

Therefore, the condition (10) is equivalent to

$$N^{\left(-\frac{1}{3}\delta^2 - \frac{1}{3}\delta + \frac{5}{6}\right)m^3 + o(m^3)} = \det(L') < e^{nm} = N^{(1-\delta)m^3 + o(m^3)}.$$

Hence we have $-\frac{1}{3}\delta^2 - \frac{1}{3}\delta + \frac{5}{6} < 1 - \delta$ for sufficiently large m. Then we have the condition for δ as follows

$$\delta < 1 - \frac{1}{\sqrt{2}} \approx 0.292. \tag{11}$$

This is the same as the Boneh-Durfee bound [2].

4 Analysis in Detail

We show that L defined in the above section is lower triangle; hence we can easily derive the determinant of the L' defined as the instantiation of L with parameters X and Y.

We need to give the detailed construction of L to prove our claim; before this, we define an index sequence $\mathbf{I}$ and a monomial sequence $\mathbf{K}$ to set an order of terms in our matrix. For fixed m and $\delta < 0.5$, we define the set $I \stackrel{\text{def}}{=} \{(i,j) \in \mathbb{Z}^2 | 0 \leq i \leq m, 0 \leq j \leq 2(1-\delta)i\}$. We respectively define $\mathbf{I}_1$ and $\mathbf{I}_2$ by the lexicographic order of (i,j) in

$\{(i,j) \in I | i \geq j\}$ and that of (j,i) in $\{(i,j) \in I | i < j\}$; we use these sequences to define the order of the vector $G_{i,j}(x,y,z)$. We further set the index sequence $\mathbf{I}$ as the concatenation of $\mathbf{I}_1$ and $\mathbf{I}_2$. For $\mathbf{I_1} = ((i_1,j_1),\ldots,(i_u,j_u))$ and $\mathbf{I_2} = ((i'_1,j'_1),\ldots,(i'_{u'},j'_{u'}))$, we construct monomial sequences $\mathbf{K}_1$ and $\mathbf{K}_2$; we use these sequences to set the monomial order in vectorization. We define the monomial sequences $\mathbf{K}_1$ and $\mathbf{K}_2$ by $\mathbf{K_1} = (x^{i_1-j_1}z^{j_1},\ldots,x^{i_u-j_u}z^{j_u})$ and $\mathbf{K_2} = (y^{j'_1-i'_1}z^{i'_1},\ldots,y^{j'_{u'}-i'_{u'}}z^{i'_{u'}})$ respectively. We also set $\mathbf{K}$ by the concatenation of $\mathbf{K}_1$ and $\mathbf{K}_2$. We use these sequences to define our L. Here we give a small example for $m = 3$ and $\delta = 0.25$; we have $\mathbf{I}_1 = ((0,0),(1,0),(1,1),(2,0),(2,1),(2,2),(3,0),(3,1),(3,2),(3,3))$ and $\mathbf{I}_2 = ((2,3),(3,4))$. By them, we have the monomial sequence $\mathbf{K}_1 = (1,x,z,x^2,xz,z^2,x^3,x^2z,xz^2,z^3)$ and $\mathbf{K}_2 = (yz^2,yz^3)$.

We state two facts for our analysis; we will use them in the proof of Lemma 3 and Lemma 4. We denote a symbol $\prec$ the order in $\mathbf{K}_1$ and $\mathbf{K}_2$.

Fact 1. *We have for the elements in* $\mathbf{K}_1$, $x^iz^j \prec x^{i'}z^{j'} \Leftrightarrow i+j < i'+j'$ *or* $[i+j = i'+j'$ *and* $j < j']$. *For the elements in* $\mathbf{K}_2$, $y^iz^j \prec y^{i'}z^{j'} \Leftrightarrow i+j < i'+j'$ *or* $[i+j = i'+j'$ *and* $j < j']$.

Fact 2. *For elements in* $\mathbf{K}$, *we have* $x^jz^i \in \mathbf{K}_1 \Leftrightarrow 0 \leq i+j \leq m$, *and* $y^jz^i \in \mathbf{K}_2 \Leftrightarrow [0 \leq i \leq m$ *and* $0 < j < (1-2\delta)i]$

Now we define our lattice L by using the polynomials $G_{i,j}(x,y,z)$ and the defined sequences; here we actually give a matrix representation of L. Our matrix is defined by the row matrix of vectors $\mathcal{V}_{\mathbf{K}}(G_{i,j})$ for $(i,j) \in I$ whose order is from $\mathbf{I}$. We divide L as follows to show its lower triangularity:

$$L = \begin{bmatrix} \mathcal{V}_{\mathbf{K}}(G_{0,0}) \\ \vdots \\ \mathcal{V}_{\mathbf{K}}(G_{m,m'}) \end{bmatrix} = \begin{array}{c} \overbrace{}^{\mathbf{K}_1} \overbrace{}^{\mathbf{K}_2} \\ \left[\begin{array}{c|c} L_{00} & L_{01} \\ \hline L_{10} & L_{11} \end{array}\right] \end{array} \begin{array}{l} \}\mathbf{I}_1 \\ \}\mathbf{I}_2 \end{array} \tag{12}$$

Here, $m' = \lfloor 2(1-\delta)m \rfloor$. Therefore, we need to show that L_{00} and L_{11} are lower triangle matrices and show that L_{01} is the zero matrix to proof the triangularity of L; of course, we need to prove that the monomials in $G_{i,j}(x,y,z)$ are contained in $\mathbf{K}$. This will be showed via the proof of Lemma 3 and Lemma 4.

Lemma 3. L_{00} *and* L_{01} *are a lower triangle matrix and the zero matrix respectively.*

Proof. Let (i_k,j_k) be k-th element in $\mathbf{I}_1$. We first show the triangularity of L_{00}; we need to show that the polynomial $G_{i,j}(x,y,z)$ can be expressed as an linear combination of the first k elements in $\mathbf{K}_1$, and show that the coefficient of $x^{i_k-j_k}z^{j_k}$ in $G_{i_k,j_k}(x,y,z)$ is not zero.

The expression of $G_{i,j}(x,y,z)$ is computed as follows by the definition (6) and the canonical replacement:

$$\begin{aligned} G_{i_k,j_k}(x,y,z) &= x^{i_k-j_k}(-1+xy+Ax)^{j_k}e^{m-j_k} \\ &= x^{i_k-j_k}(z+Ax)^{j_k}e^{m-j_k} = \sum_{\ell=0}^{j_k} a_\ell x^{i_k-\ell}z^\ell. \end{aligned}$$

where a_ℓ are certain integers.

Thus, by the Fact 1, $x^{i_k-j_k}z^{j_k}$, this is the k-th element in $\mathbf{K}_1$, is the most right non-zero element in $\mathcal{V}_{\mathbf{K}}(G_{i_k,j_k})$ in the order $\prec$; this also corresponds to the k-th diagonal element in our matrix. Hence the non-zero elements in $\mathbf{b}_{i_k,j_k}$ are on the diagonal position or its left; This shows that L_{00} is a lower triangle matrix. It is clear that L_{01} is the zero-matrix since the polynomial $G_{i_k,j_k}(x,y,z)$ for $(i_k,j_k) \in \mathbf{I}_1$ does not have a monomial of type $z^{j'}y^{i'}$. □

Lemma 4. *L_{11} is a lower triangle matrix.*

We omit the proof of this lemma. We can prove this by a similar argument in Lemma 3.

Now we can easily compute the determinant of L'; since L and its instantiation L' are lower triangle matrices by combining Lemma 3 and Lemma 4. We have from the expressions, the diagonal elements in L' corresponding to $G_{i,j}(x,y,z)$ are $e^{m-j}X^{i-j}Z^j$ for $(i,j) \in \mathbf{I}_1$, and $e^{m-i}Y^{j-i}Z^i$ for $(i,j) \in \mathbf{I}_2$ respectively. Hence by using the approximations $e \approx N, X \approx N^{\delta}, Y \approx N^{0.5}$ and $Z = \sqrt{X^2Y^2+1} \approx N^{\delta+0.5}$, we have

$$\begin{aligned} \det(L_{00}) &= e^{m(m+1)(m+2)/3}X^{m(m+1)(m+2)/6}Y^{m(m+1)(m+2)/6} \\ &= N^{\left(\frac{5}{12}+\frac{1}{3}\delta\right)m^3+o(m^3)}, \\ \det(L_{11}) &= e^{(1-2\delta)m^3/6+o(m^3)}Y^{(1-2\delta)^2m^3/6+o(m^3)}Z^{(1-2\delta)m^3/3+o(m^3)} \\ &= N^{\left(-\frac{1}{3}\delta^2-\frac{2}{3}\delta+\frac{5}{12}\right)m^3+o(m^3)}, \end{aligned} \tag{13}$$

and hence

$$\det(L) = \det(L_{00}) \cdot \det(L_{11}) = N^{\left(-\frac{1}{3}\delta^2-\frac{1}{3}\delta+\frac{5}{6}\right)m^3+o(m^3)}.$$

On the other hand, the dimension of the matrix is $n = |\mathbf{I}| = (1-\delta)m^2 + o(m^2)$. Thus we have $e^{nm} = N^{(1-\delta)m^3+o(m^3)}$. Therefore as explained in the previous section, we can derive the bound $\delta < 0.292$ by using these values.

5 On the Case of Repetitive Secret Key

In this section we give an application of our simple analysis; we propose a new RSA assumption which we named repetitive secret key.

We assume that the secret key is the repeat of r short bit string d_0 with binary length ℓ. We let $\beta = \log_N d_0$, that is, the rough ratio of the bit-length of N to that of d_0. Hence this is close to $\ell/\log_2 N$.

In this situation we derive the target equation by following the argument in [2]; our equation is

$$f_{\mathrm{rep}}(x,y) = -1 + x(y+A) \pmod{eR} \tag{14}$$

where $R = 1 + 2^{\ell} + \cdots + 2^{(r-1)\ell}$. We notice that the difference between (1) and (14) is only the modulo. To solve the equation (14), we consider the lattice based attack and we can construct a lattice with lower triangle representation by the technique in this paper.

Hence the determinant of constructed matrix is easily compute by substituting the approximations $e \approx N,\ R \approx N^{(r-1)\beta}, X \approx N^{r\beta},\ Y \approx N^{0.5},$ and $Z = \sqrt{X^2Y^2+1} \approx N^{r\beta+0.5}$, for (13). We further consider the Howgrave-Graham condition $\det(L')^{1/n} < (eR)^m$, where n and m are explained as the outline section.

Finally we obtain the condition for recovering the repetitive secret key as follows.

$$\beta < \frac{r+3-\sqrt{r^2+6r+1}}{4}. \tag{15}$$

For $r = 1$, this is equivalent to the result of Boneh and Durfee.

6 Conclusion

In this paper we study the lattice based attack for RSA with short secret key. We give the new simple analysis to obtain their bound $\delta < 0.292$. One important advantage of our technique is that it does not require any technical method or involved calculation that are necessary in the original technique. We hope that our analysis technique will be applicable in other situations of the lattice based attack.

References and Notes

[1] Y. Aono, Simplification of the lattice based attack of Boneh and Durfee for RSA cryptoanalysis, Tokyo Institute of Technology Dept. of Math. and Comp. Sciences Research Reports, C-263, available online at `http://www.is.titech.ac.jp/research/research-report/C/C-263.pdf`, 2009.

[2] D. Boneh and G. Durfee, Cryptanalysis of RSA with private key d less than $N^{0.292}$, *IEEE Transactions on Information Theory*, vol. 46, No. 4, pp. 1339-1349, 2000.

[3] J. Blömer and A. May, Low Secret Exponent RSA Revisited in *CaLC 2001*, LNCS, vol. 2146, pp. 4-19, 2001.

[4] D. Coppersmith, Finding a small root of a univariate modular equation, in *Proc. of EUROCRYPT* 1996, LNCS, vol. 1070, pp. 155-165, 1996.

[5] D. Cox, J. Little, D. O'Shea, Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, Springer-Verlag, New York, 2007.

[6] M. Ernst, E. Jochemsz, A. May, and B. Weger Partial key exposure attacks on RSA up to full size exponents, in *Proc. of EUROCRYPT'05*, LNCS, vol. 3494, pp. 371-386, 2005.

[7] A. D. Healy, Resultants, Resolvents and the Computation of Galois Groups, Available online at `http://www.alexhealy.net/papers/math250a.pdf`.

[8] N. Howgrave-Graham, Finding small roots of univariate modular equations revisited, in *Proc. of Cryptography and Coding*, LNCS, vol. 1355, pp. 131-142, 1997.

[9] E. Jochemz and A. May, A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants, in *Advances in Cryptology (Asiacrypt 2006)* LNCS, vol. 4284, pp. 267-282, 2006.

[10] A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász Factoring polynomials with rational coefficients, *Mathematische Annalen*, vol. 261, No. 4, pp. 515-534, 1982.

[11] P. Nguyen and D. Stehlé, Floating-Point LLL revisited, in *Proc. of EUROCRYPT 2005*, LNCS, vol. 3494, pp. 215-233, 2006.

[12] P. Nguyen and D. Stehlé, Floating-Point LLL (Full version), available online at `ftp://ftp.di.ens.fr/pub/users/pnguyen/FullL2.pdf`.

[13] V. Shoup, NTL: A Library for doing Number Theory, available online at `http://www.shoup.net/ntl/index.html`.

On the simplest quartic fields and related Thue equations

Akinari Hoshi

Department of Mathematics, Rikkyo University
3–34–1 Nishi-Ikebukuro Toshima-ku Tokyo, 171–8501, Japan
hoshi@rikkyo.ac.jp

Abstract

Let K be a field of char $K \neq 2$. For $a \in K$, we give an explicit answer to the field isomorphism problem of the simplest quartic polynomial $X^4 - aX^3 - 6X^2 + aX + 1$ over K. From this result, over an infinite field K, we see that the polynomial gives the same splitting field over K for infinitely many values a of K. We also see by Siegel's theorem for curves of genus zero that only finitely many algebraic integers $m \in \mathcal{O}_K$ in a number field K may give the same splitting field. By applying the result over the field $\mathbb{Q}$ of rational numbers, we establish a correspondence between primitive solutions to the parametric family of quartic Thue equations

$$X^4 - mX^3Y - 6X^2Y^2 + mXY^3 + Y^4 = c,$$

where $m \in \mathbb{Z}$ is a rational integer and c is a divisor of $4(m^2 + 16)$, and isomorphism classes of the simplest quartic fields.

1 Introduction and main results

Let K be a field of char $K \neq 2$ and $K(s)$ the rational function field over K with variable s. We take the simplest quartic polynomial

$$f_s(X) := X^4 - sX^3 - 6X^2 + sX + 1 \in K(s)[X]$$

whose Galois group $\mathrm{Gal}_{K(s)} f_s(X)$ over $K(s)$ is isomorphic to the cyclic group C_4 of order four. We note that $\mathrm{disc}_X(f_s(X)) = 4(s^2+16)^3$. In the case where $K = \mathbb{Q}$, for $a \in \mathbb{Z}\backslash\{0, \pm 3\}$, the polynomials $f_a(X)$ are irreducible over $\mathbb{Q}$ with $\mathrm{Gal}_{\mathbb{Q}} f_a(X) \cong C_4$ and the splitting fields $\mathrm{Spl}_{\mathbb{Q}} f_a(X)$ of $f_a(X)$ over $\mathbb{Q}$ are called the simplest quartic fields (cf. [Gra77], [Gra87]). We also see $\mathrm{Spl}_K f_a(X) = \mathrm{Spl}_K f_{-a}(X)$. We consider the field isomorphism problem of $f_s(X)$, i.e. for a fixed $a \in K$, whether $b \in K$ gives the same splitting field over K as $\mathrm{Spl}_K f_a(X) = \mathrm{Spl}_K f_b(X)$. In Section 3, we will give an explicit answer of the field isomorphism problem of $f_s(X)$ over K (cf. the simplest cubic case [Mor94], [Cha96], [HM09a], [H]).

Theorem 1.1. *Let K be a field of char $K \neq 2$ and $f_a(X) = X^4 - aX^3 - 6X^2 + aX + 1 \in K[X]$ for $a \in K$. For $a, b \in K$ with $a \neq \pm b$ and $(a^2 + 16)(b^2 + 16) \neq 0$, the following three conditions are equivalent*:
(i) *the splitting fields of $f_a(X)$ and of $f_b(X)$ over K coincide*;
(ii) *the polynomial $f_A(X)$ splits completely into four linear factors over K for $A = A_1$ or $A = A_2$ where*

$$A_1 = \frac{ab + 16}{-a + b} \quad \text{and} \quad A_2 = \frac{ab - 16}{a + b};$$

2000 *Mathematics Subject Classification.* Primary 11D25, 11D59, 11R16, 12F10.
Key words and phrases. Field isomorphism problem, simplest quartic fields, quartic Thue equations.
This work was partially supported by Rikkyo University Special Fund for Research.

(iii) *there exists* $z \in K$ *such that*

$$B = a + \frac{(a^2+16)z(z+1)(z-1)}{f_a(z)}$$

where $B = b$ *or* $B = -b$.

Moreover if $\mathrm{Gal}_K f_a(X) \cong C_4$ (*resp.* $\mathrm{Gal}_K f_a(X) \cong C_2$ *or* $\{1\}$) *then* (ii) *occurs for only one of* A_1 *and* A_2 (*resp. for both of* A_1 *and* A_2) *and* (iii) *occurs for only one of* b *and* $-b$ (*resp. for both of* b *and* $-b$).

Note that the equivalence of the conditions (i) and (iii) is valid also for $a = \pm b$.

By Theorem 1.1, for a fixed $a \in K$ with $a^2+16 \neq 0$, we have $\mathrm{Spl}_K f_b^{S_3}(X) = \mathrm{Spl}_K f_a^{S_3}(X)$ where b is given as in Theorem 1.1 (iii) for arbitrary $z \in K$ with $f_a(z) \neq 0$ and $b^2+16 \neq 0$. Hence we have the following:

Corollary 1.2. *Let* K *be an infinite field of char* $K \neq 2$. *For a fixed* $a \in K$ *with* $a^2+16 \neq 0$, *there exist infinitely many* $b \in K$ *such that* $\mathrm{Spl}_K f_a(X) = \mathrm{Spl}_K f_b(X)$.

In contrast with Corollary 1.2, by applying Siegel's theorem for curves of genus zero (cf. [Lan78, Theorem 6.1], [Lan83, Chapter 8, Section 5]) to Theorem 1.1, we get:

Corollary 1.3. *Let* K *be a number field and* $\mathcal{O}_K$ *the ring of integers in* K. *For* $a \in \mathcal{O}_K$ *with* $a^2+16 \neq 0$, *there exist only finitely many integers* $b \in \mathcal{O}_K$ *such that* $\mathrm{Spl}_K f_a(X) = \mathrm{Spl}_K f_b(X)$.

We treat the case of $K = \mathbb{Q}$ and $a = m \in \mathbb{Z}$. We get an application of Theorem 1.1 to a related family of quartic Thue equations as follows.

Consider the parametric family of quartic Thue equations

$$F_m(X,Y) := X^4 - mX^3Y - 6X^2Y^2 + mXY^3 + Y^4 = c$$

for m, $c \in \mathbb{Z}$ with $c \neq 0$. Note that $f_m(X) = F_m(X,1)$. The equation $F_m(X,Y) = c$ has the following solutions

$$F_m(0,\pm e) = F_m(\pm e,0) = e^4, \qquad F_m(\mp e,\pm e) = F_m(\pm e,\pm e) = -4e^4.$$

We call such solutions (x,y) to $F_m(x,y) = c$ with $xy(x+y)(x-y) = 0$ the *trivial* solutions.

For $c \in \{\pm 1, \pm 4\}$, Lettl-Pethö [LP95] and Chen-Voutier [CV97] gave a complete solution to Thue equation $F_m(X,Y) = c$ independently. For $c \in \{\pm 1, \pm 4\}$ and $m \geq 0$, all solutions to $F_m(X,Y) = c$ are given by eight trivial solutions for arbitrary $m \geq 0$ and additionally

$$F_1(\mp 2,\pm 1) = F_1(\pm 1,\pm 2) = -1, \qquad F_1(\pm 3,\pm 1) = F_1(\mp 1,\pm 3) = 4,$$
$$F_4(\mp 3,\pm 2) = F_4(\pm 2,\pm 3) = 1, \qquad F_4(\pm 5,\pm 1) = F_4(\mp 1,\pm 5) = -4.$$

Note that if $(x,y) \in \mathbb{Z}^2$ is a solution to $F_m(x,y) = c$ then four pairs $\pm(x,y)$, $\pm(y,-x)$ are also solutions because $F_m(X,Y)$ is invariant under the action $X \longmapsto Y \longmapsto -X$.

In [LPV99], Lettl-Pethö-Voutier showed that for $m \geq 58$, the only primitive solutions $(x,y) \in \mathbb{Z}^2$, i.e. $\gcd(x,y) = 1$, of the Thue inequality $|F_m(x,y)| \leq 6m+7$ with $|x| \leq y$ are trivial solutions $(0,1)$, $(\pm 1,1)$ and non-trivial solutions $(\pm 1,2)$. We note that $F_m(\pm 1,2) = \pm 6m - 7$.

Put $L_m := \mathrm{Spl}_{\mathbb{Q}} f_m(X)$ for $m \in \mathbb{Z}$. We give the following correspondence between integer solutions to $F_m(X,Y) = c$ and isomorphism classes of the simplest quartic fields L_m.

Theorem 1.4. *Let $m \in \mathbb{Z} \setminus \{0, \pm 3\}$ and $L_m = \mathrm{Spl}_{\mathbb{Q}} f_m(X)$. There exists an integer $n \in \mathbb{Z} \setminus \{\pm m\}$ such that $L_n = L_m$ if and only if there exists non-trivial solution $(x, y) \in \mathbb{Z}^2$, i.e. $xy(x+y)(x-y) \neq 0$, to the quartic Thue equation*

$$(*) \qquad F_m(x, y) = c$$

where c is a divisor of $4(m^2+16)$. Moreover integers m, n and solutions $(x, y) \in \mathbb{Z}^2$ to $(*)$ can be chosen to satisfy the equation

$$(**) \qquad N = m + \frac{(m^2+16)xy(x+y)(x-y)}{F_m(x, y)}$$

where either $N = n$ or $N = -n$, and the equation $(**)$ occurs for only one of $N = n$ and $N = -n$.

The assumption $m \neq 0, \pm 3$ ensures that $f_m(X)$ is irreducible over $\mathbb{Q}$, $\mathrm{Gal}_{\mathbb{Q}} f_m(X) \cong C_4$ and the equality $(**)$ holds for only one of $N = n$ and $N = -n$. This phenomenon comes from the group theoretical reason (see Section 2). Indeed, in the case of $m = \pm 3$, $f_{\pm 3}(X) = (X^2 \pm X - 1)(X^2 \mp 4X - 1)$ and the equation $(**)$ occurs for both of $N = 3$ and $N = -3$. Thus non-trivial solutions $(x, y) \in \mathbb{Z}^2$ to $(*)$ which satisfy $(**)$ for $N = -m$ exist (cf. Theorem 1.1). Hence the assumption $m \neq 0, \pm 3$ also ensures that a non-trivial solution (x, y) to $(*)$ corresponds $N \in \mathbb{Z} \setminus \{\pm m\}$ via $(**)$.

If there exists an integer $n \in \mathbb{Z} \setminus \{\pm m\}$ such that $L_n = L_m$, then we may choose a primitive solution $(x, y) \in \mathbb{Z}^2$ to $(*)$ with $(x, y) \equiv (1, 0) \pmod 2$. Then four solutions $\pm(x, y)$, $\pm(y, -x)$ to $(*)$ for $c = d$ and four solutions $\pm(x', y')$, $\pm(y', -x')$ to $(*)$ for $c = -4d$ are primitive, where $(x', y') = (x - y, x + y)$ and d is an odd divisor of $m^2 + 16$, and only these eight primitive solutions satisfy $(**)$ for the same N as in Theorem 1.4.

Corollary 1.5. *For $m \in \mathbb{Z} \setminus \{0, \pm 3\}$, let $\mathcal{N}$ be the number of primitive solutions $(x, y) \in \mathbb{Z}^2$ with $xy(x+y)(x-y) \neq 0$ to $F_m(x, y) = c$ where c is a divisor of $4(m^2+16)$. Then we have*

$$\#\Big\{ n \in \mathbb{Z} \setminus \{\pm m\} \;\Big|\; L_n = L_m,\ n > 0 \Big\} = \frac{\mathcal{N}}{8}$$

where $L_m = \mathrm{Spl}_{\mathbb{Q}} f_m(X)$. In particular, if there does not exist $n \in \mathbb{Z} \setminus \{\pm m\}$ with $L_n = L_m$ then $F_m(x, y) = c$ where c is a divisor of $4(m^2+16)$ has only trivial solutions $(x, y) \in \mathbb{Z}^2$ with $xy(x+y)(x-y) = 0$.

2 Preliminaries

In order to prove Theorem 1.1, we recall known results of the resolvent polynomials which are fundamental tools in the computational aspects of Galois theory (cf. [Coh93], [Ade01]). We intend to explain how to get an answer of the field intersection problem of $f_s(X) = X^4 - sX^3 - 6X^2 + sX + 1$, i.e. for $a, b \in K$ how to determine the intersection of $\mathrm{Spl}_K f_a(X)$ and $\mathrm{Spl}_K f_b(X)$ (cf. [HM09a], [HM09b]). An answer of the field isomorphism problem (Theorem 1.1) may be obtained as the special case of the field intersection problem.

Let $\overline{K}$ be a fixed algebraic closure of a field K. Let $f(X) := \prod_{i=1}^{m} (X - \alpha_i) \in K[X]$ be a separable polynomial of degree m with some fixed order of the roots $\alpha_1, \ldots, \alpha_m \in \overline{K}$. By resolvent polynomials with suitable invariants, we may determine the Galois group of the polynomial $f(X)$ over K as follows.

Let $R := K[x_1, \ldots, x_m]$ be the polynomial ring over K with m variables $x_1, \ldots, x_m$. For $\Theta \in R$, we take a surjective homomorphism $\omega_f : R \to k(\alpha_1, \ldots, \alpha_m)$, $\Theta(x_1, \ldots, x_m) \mapsto \Theta(\alpha_1, \ldots, \alpha_m)$, which is called the specialization map. The kernel of ω_f is the ideal $I_f := \{\Theta \in R \mid \Theta(\alpha_1, \ldots, \alpha_m) = 0\}$ in R.

Let S_m be the symmetric group of degree m. We extend the action of S_m on m letters $\{1, \ldots, m\}$ to that on R by $\pi(\Theta(x_1, \ldots, x_m)) := \Theta(x_{\pi(1)}, \ldots, x_{\pi(m)})$. We define the Galois group of $f(X)$ over K by $\mathrm{Gal}(f/K) := \{\pi \in S_m \mid \pi(I_f) \subseteq I_f\}$. Then the Galois group of the splitting field $\mathrm{Spl}_K f(X)$ of $f(X)$ over K is isomorphic to $\mathrm{Gal}(f/K)$. If we take another ordering of roots $\alpha_{\pi(1)}, \ldots, \alpha_{\pi(m)}$ of $f(X)$ for some $\pi \in S_m$, the corresponding realization of $\mathrm{Gal}(f/K)$ is conjugate in S_m. Hence, for arbitrary ordering of the roots of $f(X)$, $\mathrm{Gal}(f/K)$ is determined up to conjugacy in S_m.

For $H \leq U \leq S_m$, an element $\Theta \in R$ is called a U-primitive H-invariant if $H = \mathrm{Stab}_U(\Theta) := \{\pi \in U \mid \pi(\Theta) = \Theta\}$. For a U-primitive H-invariant Θ, the polynomial

$$\mathcal{RP}_{\Theta,U}(X) = \prod_{\overline{\pi} \in U/H} (X - \pi(\Theta)) \in R^U[X]$$

where $\overline{\pi}$ runs through the left cosets of H in U, is called the *formal* U-relative H-invariant resolvent by Θ. The polynomial

$$\mathcal{RP}_{\Theta,U,f}(X) := \omega_f(\mathcal{RP}_{\Theta,U}(X))$$

is called the U-relative H-invariant resolvent of f by Θ. The following theorem is fundamental in the theory of resolvent polynomials (see e.g. [Ade01, p.95]).

Theorem 2.1. *Let $G = \mathrm{Gal}(f/K)$, $H \leq U \leq S_m$ be a tower of finite groups with $G \leq U$ and Θ a U-primitive H-invariant. Suppose that $\mathcal{RP}_{\Theta,U,f}(X) = \prod_{i=1}^{l} h_i^{e_i}(X)$ gives the decomposition of $\mathcal{RP}_{\Theta,U,f}(X)$ into a product of powers of distinct irreducible polynomials $h_i(X), i = 1, \ldots, l$, in $K[X]$. Then we have a bijection*

$$\begin{array}{rcl} G\backslash U/H & \longrightarrow & \{h_1^{e_1}(X), \ldots, h_l^{e_l}(X)\} \\ G\,\pi\,H & \longmapsto & h_\pi(X) = \displaystyle\prod_{\tau H \subseteq G\,\pi\,H} \big(X - \omega_f(\tau(\Theta))\big) \end{array}$$

where the product runs through the left cosets τH of H in U contained in $G\,\pi\,H$, that is, through $\tau = \pi_\sigma \pi$ where π_σ runs a system of representative of the left cosets of $G \cap \pi H \pi^{-1}$; each $h_\pi(X)$ is irreducible or a power of an irreducible polynomial with $\deg(h_\pi(X)) = |G\,\pi\,H|/|H| = |G|/|G \cap \pi H \pi^{-1}|$.

Corollary 2.2. *If $G \leq \pi H \pi^{-1}$ for some $\pi \in U$ then $\mathcal{RP}_{\Theta,U,f}(X)$ has a linear factor over K. Conversely, if $\mathcal{RP}_{\Theta,U,f}(X)$ has a non-repeated linear factor over K then there exists $\pi \in U$ such that $G \leq \pi H \pi^{-1}$.*

When $\mathcal{RP}_{\Theta,U,f}(X)$ is not squarefree, there exists a suitable Tschirnhausen transformation $\hat{f}$ of f over K such that $\mathcal{RP}_{\Theta,U,\hat{f}}(X)$ is squarefree (cf. [Gir83], [Coh93, Alg. 6.3.4]).

We apply Theorem 2.1 to the cyclic quartic case. Let $f^1(X) \in K[X]$ and $f^2(X) \in K[X]$ be separable quartic polynomials over K respectively. We put $f(X) := f^1(X)f^2(X)$, $G_1 := \mathrm{Gal}(f^1/K)$, $G_2 := \mathrm{Gal}(f^2/K)$ and $G := \mathrm{Gal}(f/K)$.

We assume that $G_1, G_2 \leq C_4$ and apply Theorem 2.1 to $m = 8$, $f(X) = f^1(X)f^2(X)$, $U = \langle\sigma\rangle \times \langle\tau\rangle$, $H = \langle\sigma\tau\rangle$ or $\langle\sigma\tau^3\rangle$ where $\sigma, \tau \in S_8$ act on $R = K[x_1, \ldots, x_8]$ by

$$\sigma : x_1 \mapsto x_2 \mapsto x_3 \mapsto x_4 \mapsto x_1, \quad \tau : x_5 \mapsto x_6 \mapsto x_7 \mapsto x_8 \mapsto x_5.$$

We put $U := \langle\sigma\rangle \times \langle\tau\rangle$. Let Θ_1 (resp. Θ_2) be a U-primitive $\langle\sigma\tau\rangle$-invariant (resp. $\langle\sigma\tau^3\rangle$-invariant). Then we have the U-relative $\langle\sigma\tau\rangle$-invariant (resp. $\langle\sigma\tau^3\rangle$-invariant) resolvent polynomial of $f(X) = f^1(X)f^2(X)$ by Θ_1 (resp. Θ_2) as

$$\mathcal{R}^i_f(X) := \mathcal{RP}_{\Theta_i,U,f}(X), \quad (i = 1, 2).$$

This kind of resolvent polynomial is also called (absolute) *multi-resolvent polynomial* (cf. [RV99], [Ren04]).

For a squarefree polynomial $\mathcal{R}(X) \in K[X]$ of degree l, we define the *decomposition type* $\mathrm{DT}(\mathcal{R})$ of $\mathcal{R}(X)$ by the partition of l induced by the degrees of the irreducible factors of $\mathcal{R}(X)$ over K. By Theorem 2.1, we get the intersection field $\mathrm{Spl}_K f^1(X) \cap \mathrm{Spl}_K f^2(X)$ via the decomposition types $\mathrm{DT}(\mathcal{R}^1_f)$ and $\mathrm{DT}(\mathcal{R}^2_f)$.

Theorem 2.3. *For $f(X) = f^1(X)f^2(X) \in K[X]$ with G_1, $G_2 \leq C_4$, we assume that $\#G_1 \geq \#G_2$ and both $\mathcal{R}^1_f(X)$ and $\mathcal{R}^2_f(X)$ are squarefree. Then the Galois group $G = \mathrm{Gal}(f/K)$ and the intersection field $\mathrm{Spl}_K f^1(X) \cap \mathrm{Spl}_K f^2(X)$ are given by the decomposition types $\mathrm{DT}(\mathcal{R}^1_f)$ and $\mathrm{DT}(\mathcal{R}^2_f)$ as Table 1 shows.*

Table 1

G_1	G_2	G		$\mathrm{DT}(\mathcal{R}^1_f)$	$\mathrm{DT}(\mathcal{R}^2_f)$
C_4	C_4	$C_4 \times C_4$	$L_1 \cap L_2 = K$	4	4
		$C_4 \times C_2$	$[L_1 \cap L_2 : K] = 2$	2, 2	2, 2
		C_4	$L_1 = L_2$	2, 2	1, 1, 1, 1
				1, 1, 1, 1	2, 2
	C_2	$C_4 \times C_2$	$L_1 \cap L_2 = K$	4	4
		C_4	$L_1 \supset L_2$	4	4
	$\{1\}$	C_4	$L_1 \supset L_2 = K$	4	4
C_2	C_2	$C_2 \times C_2$	$L_1 \cap L_2 = K$	2, 2	2, 2
		C_2	$L_1 = L_2$	1, 1, 1, 1	1, 1, 1, 1
	$\{1\}$	C_2	$L_1 \supset L_2$	2, 2	2, 2
$\{1\}$	$\{1\}$	$\{1\}$	$L_1 = L_2 = K$	1, 1, 1, 1	1, 1, 1, 1

We checked the decomposition types $\mathrm{DT}(\mathcal{R}^i_f)$, $(i = 1, 2)$, on Table 1 using the computer algebra system GAP [GAP].

Now we get an answer of the field isomorphism problem of $f_s(X) = X^4 - sX^3 - 6X^2 + sX + 1$ via multi-resolvent polynomials $\mathcal{R}^i_{f_{a,b}}(X) := \mathcal{RP}_{\Theta_i,\langle\sigma\rangle\times\langle\tau\rangle,f_{a,b}}$, $(i = 1, 2)$, where $f_{a,b}(X) := f_a(X)f_b(X)$, as the special case of Theorem 2.3.

Theorem 2.4. *For $a, b \in K$ with $(a^2+16)(b^2+16) \neq 0$, we assume that both $\mathcal{R}^1_{f_{a,b}}(X)$ and $\mathcal{R}^2_{f_{a,b}}(X)$ are squarefree. Then two splitting fields of $f_a(X)$ and of $f_b(X)$ over K coincide if and only if $\mathcal{R}^1_{f_{a,b}}(X)$ or $\mathcal{R}^2_{f_{a,b}}(X)$ splits completely into four linear factors over K.*

3 Proof of Theorem 1.1

We give an explicit answer to the field intersection problem of $f_s(X)$ via suitable invariants Θ_1 and Θ_2. As the special case, we obtain Theorem 1.1.

Let $K(z)$ be the rational function field over K and σ a K-automorphism of $K(z)$ of order four which is defined by

$$\sigma : z \mapsto \frac{z-1}{z+1} \mapsto -\frac{1}{z} \mapsto -\frac{z+1}{z-1} \mapsto z.$$

We consider the fixed field $K(z)^{\langle\sigma\rangle}$ and the C_4-extension $K(z)/K(z)^{\langle\sigma\rangle}$. Then we get

$$f_s(X) = \prod_{x\in \mathrm{Orb}_{\langle\sigma\rangle}(z)} \Big(X - x\Big) = \Big(X - z\Big)\Big(X - \frac{z-1}{z+1}\Big)\Big(X + \frac{1}{z}\Big)\Big(X + \frac{z+1}{z-1}\Big)$$
$$= X^4 - sX^3 - 6X^2 + sX + 1 \quad \text{with} \quad s = \frac{z^4 - 6z^2 + 1}{z(z^2-1)},$$

as the generating polynomial of the field extension $K(z)/K(z)^{\langle\sigma\rangle}$. It follows that $K(z)^{\langle\sigma\rangle} = K(s)$ and the Galois group of the polynomial $f_s(X)$ over $K(s)$ is isomorphic to C_4.

We also take another rational function field $K(w)$ over K with indeterminate w, a K-automorphism τ of $K(w)$ defined by

$$\tau : w \ \mapsto \ \frac{w-1}{w+1} \ \mapsto \ -\frac{1}{w} \ \mapsto \ -\frac{w+1}{w-1} \ \mapsto w$$

and $f_t(X) = X^4 - tX^3 - 6X^2 + tX + 1$ with $t = (w^4 - 6w^2 + 1)/(w(w^2-1))$, by the similar manner of $K(z)$, σ and $f_s(X)$.

Put $U := \langle\sigma\rangle \times \langle\tau\rangle$. Then the field $K(z,w)$ is $(C_4 \times C_4)$-extension of $K(z,w)^U = K(s,t)$.

In order to apply Theorem 2.4, we should find suitable U-primitive $\langle\sigma\tau\rangle$-invariant Θ_1 and U-primitive $\langle\sigma\tau^3\rangle$-invariant Θ_2. The following is the key lemma of this paper:

Lemma 3.1. *Let $U = \langle\sigma\rangle \times \langle\tau\rangle$. We take*

$$\Theta_1 := \frac{zw+1}{-z+w} \quad \textit{and} \quad \Theta_2 := \frac{zw-1}{z+w}.$$

Then the following assertions hold:
(i) *the element Θ_1 is a U-primitive $\langle\sigma\tau\rangle$-invariant*;
(ii) *the element Θ_2 is a U-primitive $\langle\sigma\tau^3\rangle$-invariant*;
(iii) *the U-orbit of Θ_i is given by the same as $\langle\sigma\rangle$-orbit of z*;

$$\mathrm{Orb}_U(\Theta_i) = \Big\{\Theta_i, \ \frac{\Theta_i - 1}{\Theta_i + 1}, \ -\frac{1}{\Theta_i}, \ -\frac{\Theta_i + 1}{\Theta_i - 1}\Big\}, \quad (i = 1, 2).$$

The multi-resolvent polynomials $\mathcal{R}^i_{f_{a,b}}(X) := \mathcal{RP}_{\Theta_i,\langle\sigma\rangle\times\langle\tau\rangle,f_af_b}(X)$, $(i = 1, 2)$, with respect to Θ_1 and Θ_2 as in Lemma 3.1 are given by

$$\mathcal{R}^1_{f_{a,b}}(X) = f_{A_1}(X) = X^4 - \frac{ab+16}{-a+b}X^3 - 6X^2 + \frac{ab+16}{-a+b}X + 1, \tag{1}$$
$$\mathcal{R}^2_{f_{a,b}}(X) = f_{A_2}(X) = X^4 - \frac{ab-16}{a+b}X^3 - 6X^2 + \frac{ab-16}{a+b}X + 1$$

where $A_1 = (ab+16)/(-a+b)$ and $A_2 = (ab-16)/(a+b)$. Note that

$$\mathrm{disc}(\mathcal{R}^1_{f_{a,b}}(X)) = \frac{4(a^2+16)^3(b^2+16)^3}{(a-b)^6}, \quad \mathrm{disc}(\mathcal{R}^2_{f_{a,b}}(X)) = \frac{4(a^2+16)^3(b^2+16)^3}{(a+b)^6}.$$

By Theorem 2.3, we get the intersection field $\mathrm{Spl}_K f_a(X) \cap \mathrm{Spl}_K f_b(X)$ via Table 1.

Theorem 3.2. *Let $\mathcal{R}^i_{f_{a,b}}(X)$, $(i = 1, 2)$ be as in* (1). *For $a, b \in K$ with $a \neq \pm b$ and $(a^2+16)(b^2+16) \neq 0$, we assume that $\#\mathrm{Gal}_K f_a(X) \geq \#\mathrm{Gal}_K f_b(X)$. Then the intersection field $\mathrm{Spl}_K f_a(X) \cap \mathrm{Spl}_K f_b(X)$ is given by the decomposition types $\mathrm{DT}(\mathcal{R}^1_{f_{a,b}})$ and $\mathrm{DT}(\mathcal{R}^2_{f_{a,b}})$ as on Table* 1 *in Theorem 2.3.*

Proof of Theorem 1.1. As the special case of Theorem 3.2, we see the conditions (i) and (ii) are equivalent (cf. also Theorem 2.4).

The condition (iii) is just a restatement of (ii). Indeed, we may check that $z \in K$ is a root of $f_{A_1}(X)$ (resp. $f_{A_2}(X)$) if and only if z satisfies the condition (iii) for $B = b$ (resp. $B = -b$). Note that if z is a root of $f_{A_i}(X)$ then $\frac{z-1}{z+1}$, $-\frac{1}{z}$, $-\frac{z+1}{z-1}$ are also roots of $f_{A_i}(X)$ for $i = 1, 2$. By Table 1 as in Theorem 2.3, if $\mathrm{Spl}_K f_a(X) \cong C_4$ (resp. C_2 or $\{1\}$) and $f_{A_i}(z)$ splits completely then $f_{A_j}(X)$ is irreducible (resp. splits completely) over K for $(i, j) = (1, 2)$ and $(2, 1)$. This completes the proof. □

4 Proof of Theorem 1.4: the correspondence

The aim of this section is to prove Theorem 1.4 which establishes the correspondence between isomorphism classes of the simplest quartic fields L_m and non-trivial solutions to quartic Thue equations $(*)$.

Proof of Theorem 1.4. We use Theorem 1.1 in the case where $K = \mathbb{Q}$.

For $m \in \mathbb{Z} \setminus \{0, \pm 3\}$, we assume that there exists an integer $n \in \mathbb{Z} \setminus \{\pm m\}$ such that $L_m = L_n$. By Theorem 1.1, there exists $z = x/y$ with $x, y \in \mathbb{Z}$ and $\gcd(x, y) = 1$ such that

$$N \;=\; m + \frac{(m^2+16)z(z+1)(z-1)}{f_m(z)} = m + \frac{(m^2+16)xy(x+y)(x-y)}{F_m(x,y)} \in \mathbb{Z}$$

where either $N = n$ or $N = -n$. By the assumption $n \neq \pm m$, we have $xy(x+y)(x-y) \neq 0$.

We will show that $c := F_m(x, y)$ divides $4(m^2 + 16)$.

Put $h(z) := (m^2 + 16)z(z + 1)(z - 1)$ and $f(z) := F_m(z, 1)$. We take the resultant

$$R_m := \mathrm{Res}_z(h(z), f(z)) = 16(m^2 + 16)^4$$

of $h(z)$ and $f(z)$ with respect to z. We see that R_m is also given by

$$R_m = \begin{vmatrix} m^2+16 & 0 & -m^2-16 & 0 & 0 & 0 & h(z)z^3 \\ 0 & m^2+16 & 0 & -m^2-16 & 0 & 0 & h(z)z^2 \\ 0 & 0 & m^2+16 & 0 & -m^2-16 & 0 & h(z)z \\ 0 & 0 & 0 & m^2+16 & 0 & -m^2-16 & h(z) \\ 1 & -m & -6 & m & 1 & 0 & f(z)z^2 \\ 0 & 1 & -m & -6 & m & 1 & f(z)z \\ 0 & 0 & 1 & -m & -6 & m & f(z) \end{vmatrix}$$
$$= 4(m^2+16)^3\Big(h(z)p(z) + f(z)q(z)\Big)$$

where $p(z) = 5z^3 - 5mz^2 - 29z + 4m$, $q(z) = -(m^2 + 16)(5z^2 - 4)$. Hence we have

$$h(z)p(z) + f(z)q(z) = 4(m^2 + 16).$$

Put $H(x, y) := (m^2 + 16)xy(x + y)(x - y)$, $P(x, y) := 5x^3 - 5mx^2y - 29xy^2 + 4my^3$, $Q(x, y) := -(m^2 + 16)y(5x^2 - 4y^2)$. Then it follows from $z = x/y$ that

$$H(x, y)P(x, y) + F_m(x, y)Q(x, y) = 4(m^2 + 16)y^7.$$

Because $F_m(x,y)$ is invariant under the action of $\sigma : x \mapsto y,\ y \mapsto -x$, we get

$$\frac{H(x,y)P(x,y)}{F_m(x,y)} + Q(x,y) = \frac{4(m^2+16)y^7}{F_m(x,y)} \in \mathbb{Z}, \quad \text{and} \quad \frac{4(m^2+16)(-x)^7}{F_m(x,y)} \in \mathbb{Z}.$$

Since x and y are relatively prime, we conclude that $c = F_m(x,y)$ divides $4(m^2+16)$.

Conversely if there exists non-trivial solution $(x,y) \in \mathbb{Z}^2$ with $xy(x+y)(x-y) \neq 0$ such that $c = F_m(x,y)$ divides $4(m^2+16)$ then we can take $N \in \mathbb{Q} \setminus \{m\}$ by

$$N = m + \frac{(m^2+16)xy(x+y)(x-y)}{F_m(x,y)}.$$

We see $N \in \mathbb{Z} \setminus \{m\}$ as follows: First we may assume that $(x,y) \not\equiv (0,0) \pmod 2$ by replacing $(x',y') = (\frac{x}{2^r}, \frac{y}{2^r})$ for $F_m(x',y') = \frac{c}{(2^r)^4} \in \mathbb{Z}$.

If $(x,y) \equiv (0,1)$ or $(1,0) \pmod 2$ then $c = F_m(x,y)$ divides m^2+16 and hence $N \in \mathbb{Z} \setminus \{m\}$, because $F_m(0,1) \equiv F_m(1,0) \equiv 1 \pmod 2$.

If $(x,y) \equiv (1,1) \pmod 2$ then $c = F_m(x,y)$ divides $(m^2+16)xy(x+y)(x-y)$ and hence $N \in \mathbb{Z} \setminus \{m\}$, because $xy(x+y)(x-y) \equiv 0 \pmod 4$.

From the assumption $m \in \mathbb{Z} \setminus \{0, \pm 3\}$, we have $\mathrm{Gal}_{\mathbb{Q}} f_m(X) \cong C_4$. Hence it follows from Theorem 1.1 (i) and (iii) that $N \in \mathbb{Z} \setminus \{0, -m\}$ and $L_m = L_N$. □

5 Primitive solutions

By the proof of Theorem 1.1 and Theorem 1.4, a non-trivial solution $(x,y) \in \mathbb{Z}^2$ to $(*)$ may be obtained as $z = x/y$ with $\gcd(x,y) = 1$ where $z \in \mathbb{Q}$ is a root of $f_A(X) = X^4 - AX^3 - 6X^2 + AX + 1$ for $A = (ab+16)/(-a+b)$ or $A = (ab-16)/(a+b)$ as in Theorem 1.1 (ii). Hence we now consider only primitive solutions $(x,y) \in \mathbb{Z}^2$, i.e. $\gcd(x,y) = 1$, to $(*)$. Note that if $(x,y) \in \mathbb{Z}^2$ is a primitive solution to $(*)$ then four pairs $\pm(x,y)$, $\pm(y,-x)$ are primitive solutions to $(*)$. Corollary 1.5 follows from the following two lemmas:

Lemma 5.1. *Put $(x',y') := (x-y, x+y)$. If $(x,y) \in \mathbb{Z}^2$ with $(x,y) \equiv (0,1)$ or $(1,0)$ (mod 2) is a primitive solution to $F_m(x,y) = c$ then c is an odd integer and $(x',y') \equiv (1,1)$ (mod 2) is a primitive solution to $F_m(x',y') = -4c$. Conversely if $(x',y') \in \mathbb{Z}^2$ with $(x',y') \equiv (1,1)$ (mod 2) is a primitive solution to $F_m(x',y') = d$ then $d = -4c$ for an odd integer c and $(x,y) = (\frac{x'+y'}{2}, \frac{-x'+y'}{2}) \equiv (0,1)$ or $(1,0)$ (mod 2) is a primitive solution to $F_m(x,y) = c$.*

Lemma 5.2. *Let $m \in \mathbb{Z} \setminus \{0, \pm 3\}$ and $L_m = \mathrm{Spl}_{\mathbb{Q}} f_m(X)$. We assume that there exists $n \in \mathbb{Z} \setminus \{\pm m\}$ such that $L_n = L_m$.*
(i) *We may choose non-trivial primitive solution $(x,y) \in \mathbb{Z}^2$ with $(x,y) \equiv (0,1)$ (mod 2) to $F_m(x,y) = d$ where d is an odd divisor of m^2+16. Then four pairs $\pm(x,y)$, $\pm(y,-x)$ are primitive solutions to $F_m(X,Y) = d$ and four pairs $\pm(x',y')$, $\pm(y',-x')$ are primitive solutions to $F_m(X,Y) = -4d$ where $(x',y') = (x-y, x+y)$.*
(ii) *All primitive solutions to $(*)$ which satisfy $(**)$ for either $N = n$ or $N = -n$ are given by the eight solutions as in* (i)*, and such solutions exist for only one of $N = n$ and $N = -n$.*

By Theorem 1.1, we obtain

$$L_1 = L_{103}, \quad L_2 = L_{22}, \quad L_4 = L_{956}$$

where $L_m = \mathrm{Spl}_{\mathbb{Q}} f_m(X)$ for $m \in \mathbb{Z}$. Hence for $m \in \{1, 2, 4, 22, 103, 956\}$, we get non-trivial eight primitive solutions to $(*)$ via Theorem 1.4 as on the following table:

Table 2

m	N	$F_m(x,y) = c$	m^2+16	$xy(x+y)(x-y)$	(x,y)
1	103	-1	17	-6	$\pm(-2,1)$, $\pm(1,2)$
1	103	4	17	24	$\pm(3,1)$, $\pm(-1,3)$
2	-22	5	20	-6	$\pm(-2,1)$, $\pm(1,2)$
2	-22	-20	20	24	$\pm(3,1)$, $\pm(-1,3)$
4	-956	1	32	-30	$\pm(-3,2)$, $\pm(2,3)$
4	-956	-4	32	120	$\pm(5,1)$, $\pm(-1,5)$
22	-2	$125 = 5^3$	$500 = 2^2 5^3$	-6	$\pm(-2,1)$, $\pm(1,2)$
22	-2	$-500 = -2^2 5^3$	$500 = 2^2 5^3$	24	$\pm(3,1)$, $\pm(-1,3)$
103	1	$-625 = -5^4$	$10625 = 5^4 17$	6	$\pm(2,1)$, $\pm(-1,2)$
103	1	$2500 = 2^2 5^4$	$10625 = 5^4 17$	-24	$\pm(-3,1)$, $\pm(1,3)$
956	-4	$28561 = 13^4$	$913952 = 2^5 13^4$	-30	$\pm(-3,2)$, $\pm(2,3)$
956	-4	$-114244 = -2^2 13^4$	$913952 = 2^5 13^4$	120	$\pm(5,1)$, $\pm(-1,5)$

6 Reducible case

In the reducible cases $m = 0, \pm 3$, $f_m(X)$ splits as $f_0(X) = (X^2 + 2X - 1)(X^2 - 2X - 1)$ and $f_{\pm 3}(X) = (X^2 \pm X - 1)(X^2 \mp 4X - 1)$ over $\mathbb{Q}$, and hence $\mathrm{Spl}_{\mathbb{Q}} f_0(X) = \mathbb{Q}(\sqrt{2})$ and $\mathrm{Spl}_{\mathbb{Q}} f_{\pm 3}(X) = \mathbb{Q}(\sqrt{5})$.

If $m = 0$, the trivial solutions correspond to $N = \pm m = 0$.

If $m = 3$, then the eight trivial solutions $\pm(0,1)$, $\pm(1,0)$ for $c = 1$ and $\pm(-1,1)$, $\pm(1,1)$ for $c = -4$ give $N = 3$ and non-trivial eight solutions $\pm(2,1)$, $\pm(-1,2)$ for $c = -25$ and $\pm(-3,1)$, $\pm(1,3)$ for $c = 100$ give $N = -3$.

Table 3

m	N	$F_m(x,y) = c$	m^2+16	$xy(x+y)(x-y)$	(x,y)
3	-3	-25	25	6	$\pm(2,1)$, $\pm(-1,2)$
3	-3	100	25	-24	$\pm(-3,1)$, $\pm(1,3)$

However we do not know other non-trivial primitive solutions to $(*)$ for $m \geq 0$ except on Table 2 and Table 3. By the correspondence as in Theorem 1.4, in order to find primitive solutions to $(*)$ we should get $L_m = L_n$ for some $m \neq \pm n$. In [HM09b, Example 5.4], we checked with the aid of computer that for integers $0 \leq m < n \leq 10^5$, $L_m = L_n$ if and only if $(m,n) \in \{(1,103), (2,22), (4,956)\}$.

References and Notes

[Ade01] C. Adelmann, *The decomposition of primes in torsion point fields*, Lecture Notes in Mathematics, 1761, Springer-Verlag, Berlin, 2001.

[Cha96] R. J. Chapman, *Automorphism polynomials in cyclic cubic extensions*, J. Number Theory **61**, 283–291, 1996.

[CV97] J. Chen, P. Voutier, *Complete solution of the Diophantine equation $X^2+1 = dY^4$ and a related family of quartic Thue equations*, J. Number Theory **62**, 71–99, 1997.

[Coh93] H. Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics, vol. 138, Springer, Heidelberg, 1993.

[GAP] The GAP Group, GAP — Groups, Algorithms, and Programming, Version 4.4.10; 2007 (http://www.gap-system.org).

[Gir83] K. Girstmair, *On the computation of resolvents and Galois groups*, Manuscripta Math. **43**, 289–307, 1983.

[Gra77] M. N. Gras, *Table numérique du nombre de classes et des unités des extensions cycliques réelles de degré 4 de* $\mathbf{Q}$, Publ. Math. Fac. Sci. Besancon, fasc 2, 1977/1978.

[Gra87] M. N. Gras, *Special units in real cyclic sextic fields*, Math. Comp. **48**, 179–182, 1987.

[H] A. Hoshi, *On correspondence between solutions of a parametric family of cubic Thue equations and isomorphic simplest cubic fields*, preprint, arXiv:0810.3374.

[HM09a] A. Hoshi, K. Miyake, *A geometric framework for the subfield problem of generic polynomials via Tschirnhausen transformation*, Number Theory and Applications: Proceedings of the International Conferences on Number Theory and Cryptography, Hindustan Book Agency, 65–104, 2009.

[HM09b] A. Hoshi, K. Miyake, *On the field intersection problem of quartic generic polynomials via formal Tschirnhausen transformation*, Comment. Math. Univ. St. Pauli **58**, 51–86, 2009.

[Lan78] S. Lang, *Elliptic curves: Diophantine analysis*, Grundlehren der Mathematischen Wissenschaften 231, Springer-Verlag, Berlin-New York, 1978.

[Lan83] S. Lang, *Fundamentals of Diophantine geometry*, Springer-Verlag, New York, 1983.

[LP95] G. Lettl, A. Pethö, *Complete solution of a family of quartic Thue equations*, Abh. Math. Sem. Univ. Hamburg **65**, 365–383, 1995.

[LPV99] G. Lettl, A. Pethö, P. Voutier, *Simple families of Thue inequalities*, Trans. Amer. Math. Soc. **351**, 1871–1894, 1999.

[Mor94] P. Morton, *Characterizing cyclic cubic extensions by automorphism polynomials*, J. Number Theory **49**, 183–208, 1994.

[RV99] N. Rennert and A. Valibouze, *Calcul de résolvantes avec les modules de Cauchy*, Experiment. Math. **8**, 351–366, 1999.

[Ren04] N. Rennert, *A parallel multi-modular algorithm for computing Lagrange resolvens*, J. Symbolic Comput. **37**, 547–556, 2004.

In-place Arithmetic for Univariate Polynomials over an Algebraic Number Field

Seyed Mohammad Mahdi Javadi[1*], Michael Monagan[2*]

[1] School of Computing Science, Simon Fraser University,
Burnaby, B.C., V5A 1S6, Canada.
sjavadi@cs.sfu.ca

[2] Department of Mathematics, Simon Fraser University,
Burnaby, B.C., V5A 1S6, Canada.
mmonagan@cecm.sfu.ca

Abstract

We present a C library of *in-place* subroutines for univariate polynomial multiplication, division and GCD over L_p where L_p is an algebraic number field L with multiple field extensions reduced modulo a machine prime p. We assume elements of L_p and L are represented using a recursive dense representation. The main feature of our algorithms is that we eliminate the storage management overhead which is significant compared to the cost of arithmetic in $\mathbb{Z}_p$ by pre-allocating the exact amount of storage needed for both the output and working storage. We give an analysis for the working storage needed for each in-place algorithm and provide benchmarks demonstrating the efficiency of our library. This work improves the performance of polynomial GCD computation over algebraic number fields.

1 Introduction

In 2002, van Hoeij and Monagan in [10] presented an algorithm for computing the monic GCD $g(x)$ of two polynomials $f_1(x)$ and $f_2(x)$ in $L[x]$ where $L = \mathbb{Q}(\alpha_1, \alpha_2, \ldots, \alpha_k)$ is an algebraic number field. The algorithm is a *modular* GCD algorithm. It computes the GCD of f_1 and f_2 modulo a sequence of primes $p_1, p_2, \ldots, p_l$ using the monic Euclidean algorithm in $L_p[x]$ and it reconstructs the rational numbers in $g(x)$ using Chinese remaindering and rational number reconstruction. The algorithm is a generalization of earlier work of Langymyr and MaCallum [5], and Encarnación [2] to treat the case where L has multiple extensions ($k > 1$). It can be generalized to multivariate polynomials in $L[x_1, x_2, \ldots, x_n]$ using evaluation and interpolation (see [4, 11]).

Monagan implemented the algorithm in Maple in 2001 and in Magma in 2003 using the *recursive dense* polynomial representation to represent elements of L, L_p, $L[x_1, \ldots, x_n]$ and $L_p[x_1, \ldots, x_n]$. This representation is generally more efficient than the distributed and recursive sparse representations for sparse polynomials. See for example the comparison by Fateman in [3]. And since efficiency in the recursive dense representation improves for dense polynomials, and elements of L are often dense, it should be a good choice for implementing arithmetic in L and also L_p.

However, we have observed that arithmetic in L_p is very slow when α_1 has low degree. Since this case often occurs in practical applications, and since over 90% of a GCD computation in $L[x]$ is typically spent in the Euclidean algorithm in $L_p[x]$, we sought to improve the efficiency of the arithmetic in L_p. One reason why this happens is because the cost

*Correspondence to: CECM, Simon Fraser University, Burnaby, BC, Canada. Tel: +1.778.782.5617

of storage management, allocating small arrays for storing intermediate polynomials of low degree can be much higher than the cost of the actual arithmetic being done in $\mathbb{Z}_p$.

Our main contribution is a library of *in-place* algorithms for arithmetic in L_p and $L_p[x]$ where L_p has one or more extensions. The main idea is to eliminate all calls to the storage manager by pre-allocating one large piece of working storage, and re-using parts of it in a computation. In Section 2 we describe the recursive dense polynomial representation for elements of $L_p[x]$. In Section 3 we present algorithms for multiplication and inversion in L_p and multiplication, division with remainder and GCD in $L_p[x]$ which are given one array of storage in which to write the output and one additional array W of working storage for intermediate results. In Section 4 we give formulae for determining the size of W needed for each algorithm. In each case the amount of working storage is linear in d the degree of L. We have implemented our algorithms in the C language in a library which includes also algorithms for addition, subtraction, and other utility routines. In Section 5 we present benchmarks demonstrating its efficiency by comparing our algorithms with the Magma ([1]) computer algebra system and we explain how to avoid most of the integer divisions by p when doing arithmetic in $\mathbb{Z}_p$ because this also significantly affects overall performance.

2 Polynomial Representation

Let $\mathbb{Q}(\alpha_1, \alpha_2, \ldots, \alpha_r)$ be our number field L. We build L as follows. For $1 \leq i \leq r$, let $m_i(z_1, \ldots, z_i) \in \mathbb{Q}[z_1, \ldots, z_i]$ be the minimal polynomial for α_i, monic and irreducible over $\mathbb{Q}[z_1, \ldots, z_{i-1}]/\langle m_1, \ldots, m_{i-1}\rangle$. Let $d_i = \deg_{z_i}(m_i)$. We assume $d_i \geq 2$. Let $L = \mathbb{Q}[z_1, \ldots, z_r]/\langle m_1, \ldots, m_r\rangle$. So L is an algebraic number field of degree $d = \prod d_i$ over $\mathbb{Q}$. For a prime p for which the rational coefficients of m_i exist modulo p, let $R_i = \mathbb{Z}_p[z_1, \ldots, z_i]/\langle \bar{m}_1, \ldots, \bar{m}_i\rangle$ where $\bar{m}_i = m_i \bmod p$ and let $R = R_r = L \bmod p$. We use the following recursive dense representation for elements of R and polynomials in $R[x]$ for our algorithms. We view an element of R_{i+1} as a polynomial with degree at most $d_{i+1} - 1$ with coefficients in R_i.

To represent a non-zero element $\beta_1 = a_0 + a_1 z_1 + \cdots + a_{d_1-1} z_1^{d_1-1} \in R_1$ we use an array A_1 of size $S_1 = d_1 + 1$ indexed from 0 to d_1, of integers (modulo p) to store β_1. We store $A_1[0] = \deg_{z_1}(\alpha_1)$ and, for $0 \leq i < d_1 : A_1[i+1] = a_i$. Note that if $\deg_{z_1}(\alpha_1) = \bar{d} < d_1 - 1$ then for $\bar{d} + 1 < j \leq d_1$, $A_1[j] = 0$. To represent the zero element of R_1 we use $A[0] = -1$.

Now suppose we want to represent an element $\beta_2 = b_0 + b_1 z_2 + \cdots + b_{d_2-1} z_2^{d_2-1} \in R_2$ where $b_i \in R_1$ using an array A_2 of size $S_2 = d_2 S_1 + 1 = d_2(d_1 + 1) + 1$. We store $A_2[0] = \deg_{z_2}(\beta_2)$ and for $0 \leq i < d_2$

$$A_2[i(d_1+1)+1 \ldots (i+1)(d_1+1)] = B_i[0 \ldots d_1]$$

where B_i is the array which represents $b_i \in R_1$. Again if $\beta_2 = 0$ we store $A_2[0] = -1$.

Similarly, we recursively represent $\beta_r = c_0 + c_1 z_r + \cdots + c_{d_r-1} z_r^{d_r-1} \in R_r$ based on the representation of $c_i \in R_{r-1}$. Let $S_r = d_r S_{r-1} + 1$ and suppose A_r is an array of size S_r such that $A_r[0] = \deg_{z_r}(\beta_r)$ and for $0 \leq i < d_r$

$$A_r[i(d_{r-1})+1 \ldots (i+1)(d_{r-1}+1)] = C_i[0 \ldots S_{r-1} - 1].$$

Note, we store the degrees of the elements of R_i in $A_i[0]$ simply to avoid re-computing them. We have

$$\prod_{i=1}^{r} d_i < S_r < \prod_{i=1}^{r} (d_i + 1), S_r \in O(\prod_{i=1}^{r} d_i).$$

Now suppose we use the array C to represent a polynomial $f \in R_i[x]$ of degree d_x in the same way. Each coefficient of f in x is an element of R_i which needs an array of size S_i, hence C must be of size

$$P(d_x, R_i) = (d_x + 1)S_i + 1.$$

Example 1. Let $r = 2$ and $p = 17$. Let $\bar{m}_1 = z_1^3 + 3$, $\bar{m}_2 = z_2^2 + 5z_1z_2 + 4z_2 + 7z_1^2 + 3z_1 + 6$, and $f = 3 + 4z_1 + (5 + 6z_1)z_2 + (7 + 8z_1 + 9z_1^2 + (10z_1 + 11z_1^2)z_2)x + 12x^2$.
The representation for f is

$$C = \boxed{2}\underbrace{\boxed{1}\boxed{1}\boxed{3}\boxed{4}\boxed{0}\boxed{1}\boxed{5}\boxed{6}\boxed{0}}_{3+4z_1+(5+6z_1)z_2}\boxed{1}\boxed{2}\boxed{7}\boxed{8}\boxed{9}\underbrace{\boxed{2}\boxed{0}\boxed{10}\boxed{11}}_{10z_1+11z_1^2}\boxed{0}\boxed{0}\boxed{12}\boxed{0}\boxed{0}\boxed{-1}\boxed{0}\boxed{0}\boxed{0}$$

Here $d_x = 2, d_1 = 3, d_2 = 2, S_1 = d_1 + 1 = 4, S_2 = d_2S_1 + 1 = 9$ and the size of the array A is $P(d_x, R_2) = (d_x + 1)S_2 + 1 = 28$.

We also need to represent the minimal polynomial $\bar{m}_i$. Let $\bar{m}_i = a_0 + a_1z_i + \ldots a_{d_i}z_i^{d_i}$ where $a_j \in R_{i-1}$. We need an array of size S_{i-1} to represent a_j so to represent $\bar{m}_i$ in the same way we described above, we need an array of size $\bar{S}_i = 1 + (d_i + 1)S_{i-1} = d_iS_{i-1} + 1 + S_{i-1} = S_i + S_{i-1}$. We define $S_0 = 1$.

We represent the set of minimal polynomials $\{\bar{m}_1, \ldots, \bar{m}_r\}$ as an Array E of size $\sum_{i=1}^{r} \bar{S}_i = \sum_{i=1}^{r} (S_i + S_{i-1}) = 1 + S_r + 2\sum_{i=1}^{r-1} S_i$ such that $E[M_i \ldots M_{i+1} - 1]$ represents m_{r-i} where $M_0 = 0$ and $M_i = \sum_{i=r-i+1}^{r} \bar{S}_i$. The minimal polynomials in Example 1 will be represented in the following figure where $E[0 \ldots 12]$ represents $\bar{m}_2$ and $E[13 \ldots 17]$ represents $\bar{m}_1$.

$$E = \underbrace{\boxed{2}\boxed{2}\boxed{6}\boxed{3}\boxed{7}\boxed{1}\boxed{4}\boxed{5}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{0}}_{\bar{m}_2}\underbrace{\boxed{3}\boxed{3}\boxed{0}\boxed{0}\boxed{1}}_{\bar{m}_1}$$

3 In-place Algorithms

In this section we design efficient in-place algorithms for multiplication, division and GCD computation of two univariate polynomials over R. We will also give an in-place algorithm for computing the inverse of an element $\alpha \in R$, if it exists. This is needed for making a polynomial monic for the monic Euclidean algorithm in $R[x]$. We assume the following utility operations are implemented.

- IP_ADD(N, A, B) and IP_SUB(N, A, B) are used for in-place addition and subtraction of two polynomials $a, b \in R_N[x]$ represented in arrays A and B.
- IP_MUL_NO_EXT is used for multiplication of two polynomials over $\mathbb{Z}_p$. A description of this algorithm is given in Section 5.1.
- IP_REM_NO_EXT is used for computing the quotient and the remainder of dividing two polynomials over $\mathbb{Z}_p$.
- IP_INV_NO_EXT is used for computing the inverse of an element in $\mathbb{Z}_p[z]$ modulo a minimal polynomial $m \in \mathbb{Z}_p[z]$.
- IP_GCD_NO_EXT is used for computing the GCD of two univariate polynomials over $\mathbb{Z}_p$ (the Euclidean algorithm, See [7]).

3.1 In-place Multiplication

Suppose we have $a, b \in R[x]$ where $R = R_{r-1}[z_r]/\langle m_r(z_r)\rangle$. Let $a = \sum_{i=0}^{d_a} a_i x^i$ and $b = \sum_{i=0}^{d_b} b_i x^i$ where $d_a = \deg_x(a)$ and $d_b = \deg_x(b)$ and Let $c = a \times b = \sum_{i=0}^{d_c} c_i x^i$ where $d_c = \deg_x(c) = d_a + d_b$. To reduce the number of divisions by $m_r(z_r)$ when multiplying $a \times b$, we use the Cauchy product rule to compute c_k as suggested in [7], that is,

$$c_k = \left[\sum_{i=\max(0,k-d_b)}^{\min(k,d_a)} a_i \times b_{k-i} \right] \mod m_r(z_r).$$

Thus the number of multiplications in $R_{r-1}[z_r]$ (in line 11) is $(d_a + 1) \times (d_b + 1)$ and the number of divisions in $R_{r-1}[z_r]$ (in line 15) is $d_a + d_b + 1$. Asymptotically, this saves about half the work.

Algorithm IP_MUL: In-place Multiplication

Input:
- N the number of field extensions.
- Arrays $A[0 \ldots \bar{a}]$ and $B[0 \ldots \bar{b}]$ representing univariate polynomials $a, b \in R_N[x]$ ($R_N = \mathbb{Z}_p[z_1, \ldots, z_N]/\langle \bar{m}_1, \ldots, \bar{m}_N\rangle$). Note that $\bar{a} = P(d_a, R_N) - 1$ and $\bar{b} = P(d_b, R_N) - 1$ where $d_a = \deg_x(a)$ and $d_b = \deg_x(b)$.
- Array $C[0 \ldots \bar{c}]$: Space needed for storing $c = a \times b = \sum_{i=0}^{d_c} c_i x^i$ where $\bar{c} = P(\deg_x(a) + \deg_x(b), R_N) - 1$.
- $E[0 \ldots e_N]$: representing the set of minimal polynomials where $e_N = S_N + 2\sum_{i=1}^{N-1} S_i$.
- $W[0 \ldots w_N]$: *the working storage* for the intermediate operations.

Output: For $0 \le k \le d_c$, c_k will be computed and stored in $C[kS_N + 1 \ldots (k+1)S_N]$.

1: Set $d_a := A[0]$ and $d_b := B[0]$.
2: **if** $d_a = -1$ or $d_b = -1$ **then** Set $C[0] := -1$ and **return.**
3: **if** $N = 0$ **then** Call IP_MUL_NO_EXT on inputs A, B and C and **return.**
4: Let $M = E[0 \ldots \bar{S}_N - 1]$ and $E' = E[\bar{S}_N \ldots e_N]$ (M *points to* $\bar{m}_N$ *in* $E[0 \ldots e_N]$).
5: Let $T_1 = W[0 \ldots t-1]$ and $T_2 = W[t \ldots 2t-1]$ and $W' = W[2t \ldots w_N]$ where $t = P(2d_N - 2, R_{N-1})$ and $d_N = M[0] = \deg_{z_N}(\bar{m}_N)$.
6: Set $d_c := d_a + d_b$ and $s_c := 1$.
7: **for** k from 0 to d_c **do**
8: Set $s_a := 1 + iS_N$ and $s_b := 1 + (k-i)S_N$.
9: Set $T_1[0] := -1$ ($T_1 = 0$).
10: **for** i from $\max(0, k - d_b)$ to $\min(k, d_a)$ **do**
11: Call IP_MUL($N-1$, $A[s_a \ldots \bar{a}]$, $B[s_b \ldots \bar{b}], T_2, E', W'$).
12: Call IP_ADD($N-1, T_1, T_2$) ($T_1 := T_1 + T_2$)
13: Set $s_a := s_a + S_N$ and $s_b := s_b - S_N$.
14: **end for**
15: Call IP_REM($N-1, T_1, M, E', W'$). (*Reduce* T_1 *modulo* $M = \bar{m}_N$).
16: Copy $T_1[0 \ldots S_N - 1]$ into $C[s_c \ldots s_c + S_N - 1]$.
17: Set $s_c := s_c + S_N$.
18: **end for**
19: Determine $\deg_x(a \times b)$: (*There might be zero-divisors*).
20: Set $s_c := s_c - S_N$.
21: **while** $d_c \ge 0$ and $C[sc] = -1$ **do** Set $d_c := d_c - 1$ and $s_c := s_c - S_N$.
22: Set $C[0] := d_c$.

The temporary variables T_1 and T_2 must be big enough to store the product of two coefficients in $a, b \in R_N[x]$. Coefficients of a and b are in $R_{N-1}[z_N]$ with degree (in z_N) at most $d_N - 1$. Hence these temporaries must be of size $P(d_N - 1 + d_N - 1, R_{N-1}) = P(2d_N - 2, R_{N-1})$.

3.2 In-place Division

The following algorithm divides a polynomial $a \in R_N[x]$ by a *monic* polynomial $b \in R_N[x]$. The remainder and the quotient of a divided by b will be stored in the array representing a hence a is destroyed by the algorithm. The division algorithm is organized differently from the normal long division algorithm which does $d_b \times (d_a - d_b + 1)$ multiplications and divisions in $R_{N-1}[z_r]$. The number of divisions by M in $R_{N-1}[z_r]$ in line 16 is reduced to $d_a + 1$ (see line 8). Asymptotically this saves half the work.

Algorithm IP_REM: In-place Remainder

Input:
- N the number of field extensions.
- Arrays $A[0 \dots \bar{a}]$ and $B[0 \dots \bar{b}]$ representing univariate polynomials $a, b \neq 0 \in R_N[x]$ $(R_N = \mathbb{Z}_p[z_1, \dots, z_N]/\langle \bar{m}_1, \dots, \bar{m}_N \rangle)$ where $d_a = \deg_x(a) \geq d_a = \deg_x(b)$. Note b must be monic and $\bar{a} = P(d_a, R_N) - 1$ and $\bar{b} = P(d_b, R_N) - 1$.
- $E[0 \dots e_N]$: representing the set of minimal polynomials where $e_N = S_N + 2\sum_{i=1}^{N-1} S_i$.
- $W[0 \dots w_N]$: *the working storage* for the intermediate operations.

Output: The remainder $\bar{R}$ of a divided by b will be stored in $A[0 \dots \bar{r}]$ where $\bar{r} = P(D_r, R_N) - 1$ and $D_r = \deg_x(\bar{R}) \leq d_b - 1$. Also let Q represent the quotient $\bar{Q}$ of a divided by b. $Q[1 \dots \bar{q}]$ will be stored in $A[1 + d_b S_N \dots \bar{a}]$ where $\bar{q} = P(d_a - d_b, R_N) - 1$.

1: Set $d_a := A[0]$ and $d_b := B[0]$.
2: **if** $d_a < d_b$ **then return**.
3: **if** $N = 0$ **then** Call IP_REM_NO_EXT on inputs A and B and **return.**
4: Set $D_q := d_a - d_b$ and $D_r := d_b - 1$.
5: Let $M = E[0 \dots \bar{S}_N - 1]$ and $E' = E[\bar{S}_N \dots e_N]$ (M *points to* $\bar{m}_N$ *in* $E[0 \dots e_N]$).
6: Let $T_1 = W[0 \dots t-1]$ and $T_2 = W[t \dots 2t-1]$ and $W' = W[2t \dots w_N]$ where $t = P(2d_N - 2, R_{N-1})$ and $d_N = M[0] = \deg_{z_N}(\bar{m}_N)$.
7: Set $s_c := 1 + d_a S_N$
8: **for** $k = d_a$ to 0 by -1 **do**
9: Copy $A[s_c \dots s_c + S_N - 1]$ into $T_1[0 \dots S_N - 1]$.
10: Set $i := \max(0, k - D_q)$, $s_b := 1 + iS_N$ and $s_a := 1 + (k - i + d_b)S_N$.
11: **while** $i \leq \min(D_r, k)$ **do**
12: Call IP_MUL($N-1, A[s_a \dots \bar{a}]$, $B[s_b \dots \bar{b}]$, T_2, E', W').
13: Call IP_SUB($N-1, T_1, T_2$) ($T_1 := T_1 - T_2$).
14: Set $s_b := s_b + S_N$ and $s_a := s_a - S_N$.
15: **end while**
16: Call IP_REM($N-1, T_1$, M, E', W') (*Reduce* T_1 *modulo* $M = \bar{m}_N$).
17: Copy $T_1[0 \dots S_N - 1]$ into $A[s_c \dots s_c + S_N - 1]$.
18: Set $s_c := s_c - S_N$.
19: **end for**
20: Set $s_c := 1 + D_r S_N$.
21: **while** $D_r \geq 0$ and $A[s_c] = -1$ **do** Set $D_r := D_r - 1$ and $s_c := s_c - S_N$.
22: Set $A[0] := D_r$.

Let arrays A and B represent polynomials a and b respectively. Let $d_a = \deg_x(a)$ and $d_b = \deg_x(b)$. Array A has enough space to store $d_a + 1$ *coefficients* in R_N plus one unit of storage to store d_a. Hence the total storage is $(d_a + 1)S_N + 1$. The remainder $\bar{R}$ is of degree at most $d_b - 1$ in x, i.e. $\bar{R}$ needs storage for d_b coefficients in R_N and one unit for the degree. Similarly the quotient $\bar{Q}$ is of degree $d_a - d_b$, hence needs storage for $d_a - d_b + 1$ coefficients and one unit for the degree. Thus the remainder and the quotient together need $d_b S_N + 1 + (d_a - d_b + 1)S_N + 1 = (d_a + 1)S_N + 2$. This means we are one unit of storage short if we want to store both $\bar{R}$ and $\bar{Q}$ in A. This is because this time we are storing two degrees for $\bar{Q}$ and $\bar{R}$. Our solution is that we will not store the degree of $\bar{Q}$. Any algorithm that

calls IP_REM and needs both the quotient and the remainder must use $\deg_x(a) - \deg_x(b)$ for the degree of $\bar{Q}$.

After applying this algorithm the remainder $\bar{R}$ will be stored in $A[0 \dots d_b S_N]$ and the quotient $\bar{Q}$ minus the degree will be stored in $A[d_b S_N \dots (d_a + 1)S_N]$. Similar to IP_MUL, the remainder operation in line 16 has been moved to outside of the main loop to let the values accumulate in T_1.

3.3 Computing (In-place) the inverse of an element in R_N

In this algorithm we assume the following in-place function:

- IP_SCAL_MUL(N, A, C, E, W): This is used for multiplying a polynomial $a \in R_N[x]$ (represented by array A) by a scalar $c \in R_N$ (represented by array C). The algorithm will multiply every coefficient of a in x by c and reduce the result modulo the minimal polynomials. It can easily be implemented using IP_MUL and IP_REM.

The algorithm computes the inverse of an element a in R_N. If the element is not invertible, then the Euclidean algorithm will compute a proper divisor of some minimal polynomial $m_i(z_i)$, a zero-divisor in R_i. The algorithm will store that zero-divisor in the space provided for the inverse and return the index i of the minimal polynomial which is reducible and has caused the zero-divisor.

Algorithm IP_INV: In-place inverse of an element in R_N

Input:
- $N \geq 1$ the number of field extensions.
- Array $A[0 \dots S_N - 1]$ representing the univariate polynomial $a \in R_N$.
- Array $I[0 \dots S_N - 1]$: Space needed for storing the inverse $a^{-1} \in R_N$.
- $E[0 \dots e_N]$ representing the set of minimal polynomials. Note $e_N = S_N + 2\sum_{i=1}^{N-1} S_i$.
- $W[0 \dots w_N]$: *the working storage* for the intermediate operations.

Output: The inverse of a (or a zero-divisor, if there exists one) will be computed and stored in I. If there is a zero-divisor, the algorithm will return the index k where $\bar{m}_k$ is the reducible minimal polynomial, otherwise it will return 0.

1: Let $M = E[0 \dots \bar{S}_N - 1]$ and $E' = E[\bar{S}_N \dots e_N]$ $(M = \bar{m}_N)$.
2: **if** $N = 1$ **then** Call IP_INV_NO_EXT on inputs A, I, E, M and W and **return.**
3: **if** $A[i] = 0$ for all $0 \leq i < N$ and $A[N] = 1$ (*Test if* $a = 1$) **then**
4: Copy A into I and **return 0.**
5: **end if**
6: Let $r_1 = W[0 \dots t-1], r_2 = W[t \dots 2t-1], s_1 = I, s_2 = W[2t \dots 3t-1], T = W[3t \dots 4t-1]$, $T' = W[4t \dots 4t + t' - 1]$ and $W' = W[4t + t' \dots w_N]$ where $t = P(d_N, R_{N-1}) - 1 = \bar{S}_N - 1$, $t' = P(2d_N - 2, R_{N-1})$ and $d_N = M[0] = \deg_{z_N}(\bar{m}_N)$.
7: Copy A and M into r_1 and r_2 respectively.
8: Set $s_2[0] := -1$ (s_2 *represents* 0).
9: Let $Z \in \mathbb{Z} :=$ IP_INV$(N-1, A + D_a S_{N-1} + 1, T, E', W')$ where $D_a = A[0] = \deg_{z_N}(a)$. ($A[D_a S_{N-1} + 1 \dots S_N - 1]$ *represents* $l = lc_{z_N}(a)$ *and* T *represents* l^{-1}.)
10: **if** $Z > 0$ **then** Copy T into I and **return** Z.
11: Copy T into s_1.
12: Call IP_SCAL_MUL(N, r_1, T, E', W') (r_1 *is made monic*).
13: **while** $r_2[0] \neq -1$ **do**
14: Set $Z =$ IP_INV$(N-1, r_2 + D_{r_2} S_{N-1} + 1, T, E', W')$ where $D_{r_2} = r_2[0] = \deg_{z_N}(r_2)$.
15: **if** $Z > 0$ **then** Copy T into I and **return** Z.
16: Call IP_SCAL_MUL(N, r_2, T, E', W') (r_2 *is made monic*).
17: Call IP_SCAL_MUL(N, s_2, T, E', W').
18: Set $D_q := \max(-1, r_1[0] - r_2[0])$.

19: Call IP_REM(N, r_1, r_2, E', W').
20: Swap the arrays r_1 and r_2. (*Interchange only the pointers*).
21: Set $t_1 := r_2[r_1[0]S_{N-1}]$ and set $r_2[r_1[0]S_{N-1}] := D_q$.
22: Call IP_MUL$(N-1, r_2[r_1[0]S_{N-1} \ldots S_N - 1], s_2, T', E', W')$.
23: Call IP_REM$(N-1, T', M, E', W')$ and then IP_SUB$(N-1, s_1, T')$. ($s_1 := s_1 - qs_2$.)
24: Set $r_2[r_1[0]S_{N-1}] := t_1$.
25: Swap the arrays s_1 and s_2. (*Interchange only the pointers*).
26: **end while**
27: **if** $r_1[i] = 0$ for all $0 \leq i < N$ and $r_1[N] = 1$ **then**
28: Copy s_1 into I ($r_1 = 1$ *and* s_1 *is the inverse*) and **return** 0.
29: **else**
30: Copy r_1 into I ($r_1 \neq 1$ *is the zero-divisor*) and **return** $N-1$ ($\bar{m}_{N-1}$ *is reducible*).
31: **end if**

As discussed in Section 3.2, IP_REM will not store the degree of the quotient of a divided by b hence in line 21 we explicitly compute and set the degree of the quotient before using it to compute $s_1 := s_1 - qs_2$ in lines 22 and 23. Here $r_2[r_1[0]S_{N-1} \ldots S_N - 1]$ is the quotient of $r_1 \div r_2$ in line 19.

3.4 In-place GCD Computation

In the following algorithm we compute the GCD of $a, b \in R_N[x]$ using the monic Euclidean algorithm. Note, since $m_i(z_i)$ may be reducible modulo p, R_N is is not necessarily a field, and therefore, the monic Euclidean algorithm may encounter a zero-divisor in R_N when calling subroutine IP_INV.

Algorithm IP_GCD: In-place GCD Computation

Input:
- N the number of field extensions.
- Arrays $A[0 \ldots \bar{a}]$ and $B[0 \ldots \bar{b}]$ representing univariate polynomials $a, b \neq 0 \in R_N[x]$ ($R_N = \mathbb{Z}_p[z_1, \ldots, z_N] / \langle \bar{m}_1, \ldots, \bar{m}_N \rangle$) where $d_a = \deg_x(a) \geq d_a = \deg_x(b)$ and $A, B \neq 0$. Note that b is monic and $\bar{a} = P(d_a, R_N) - 1$ and $\bar{b} = P(d_b, R_N) - 1$.
- $E[0 \ldots e_N]$: representing the set of minimal polynomials where $e_N = S_N + 2\sum_{i=1}^{N-1} S_i$.
- $W[0 \ldots w_N]$: *the working storage* for the intermediate operations.

Output: If a zero-divisor is encountered, it will be stored in A and the index of the reducible minimal polynomial will be returned. Otherwise the monic GCD $g = \gcd(a, b)$ will be stored in A and 0 will be returned. Also, B is destroyed.

1: **if** $N = 0$ **then** CALL IP_GCD_NO_EXT on inputs A and B and **return** 0.
2: Set $d_a := A[0]$ and $d_b := B[0]$.
3: Let r_1 and r_2 point to A and B respectively.
4: Let $I = W[0 \ldots t-1]$ and $W' = W[t \ldots w_N]$ where $t = \bar{S}_N - 1 = S_N + S_{N-1} - 1$.
5: Let Z be the output of IP_INV$(N, r_1 + r_1[0]\, S_N + 1, I, E, W')$.
6: **if** $Z > 0$ **then** Copy I into A and **return** Z.
7: Call IP_SCAL_MUL(N, r_1, I, E, W').
8: **while** $r_2[0] \neq -1$ **do**
9: Let Z be the output of IP_INV$(N, r_2 + r_2[0]\, S_N + 1,\ I,\ E, W')$.
10: **if** $Z > 0$ **then** Copy I into A and **return** Z.
11: Call IP_SCAL_MUL(N, r_2, I, E, W') (*make* r_2 *monic*).
12: Call IP_REM(N, r_1, r_2, E, W') (*the remainder of* $r_1 \div r_2$ *is in* r_1).
13: Swap r_1 and r_2 (*interchange pointers*).
14: **end while**
15: Copy r_1 into A.
16: **return** 0.

Similar to the algorithm IP_INV, if there exists a zero-divisor, i.e. the leading coefficient of one of the polynomials in the polynomial remainder sequence is not invertible, in steps 6 and 10 the algorithm stores the zero-divisor in the space provided for a and returns Z the index of the minimal polynomial which is reducible and has caused the zero-divisor.

4 Working Space

In this section we will determine recurrences for the exact amount of working storage w_N needed for each operation introduced in the previous section. Recall that $d_i = \deg_{z_i}(\bar{m}_i)$ is the degree of the ith minimal polynomial which we may assume is at least 2. Also S_i is the space needed to store an element in R_i and we have $S_{i+1} = d_{i+1}S_i + 1$ and $S_1 = d_1 + 1$.

Lemma 2. $S_N > 2S_{N-1}$ for $N > 1$.

Proof. We have $S_N = d_N S_{N-1} + 1$ where $d_N = \deg_{z_N}(\bar{m}_N)$. Since $d_N \geq 2$ we have $S_N \geq 2S_{N-1} + 1 \Rightarrow S_N > 2S_{N-1}$. □

Lemma 3. $\sum_{i=1}^{N-1} S_i < S_N$ for $N > 1$.

Proof. (by induction on N). For $N = 2$ we have $\sum_{i=1}^{1} S_i = S_1 < S_2$. For $N = k + 1 \geq 2$ we have $\sum_{i=1}^{k} S_i = S_k + \sum_{i=1}^{k-1} S_i$. By induction we have $\sum_{i=1}^{k-1} S_i < S_k$ hence $\sum_{i=1}^{k} S_i < S_k + S_k = 2S_k$. Using Lemma 2 we have $2S_k < S_{k+1}$ hence $\sum_{i=1}^{k} S_i < 2S_k < S_{k+1}$ and the proof is complete. □

Corollary 4. $\sum_{i=1}^{N} S_i < 2S_N$ for $N > 1$.

Lemma 5. $P(2d_N - 2, R_{N-1}) = 2S_N - S_{N-1} - 1$ for $N > 1$.

Proof. We have $P(2d_N - 2, R_{N-1}) = (2d_N - 1)S_{N-1} + 1 = 2d_N S_{N-1} - S_{N-1} + 1 = 2(d_N S_{N-1} + 1) - S_{N-1} - 1 = 2S_N - S_{N-1} - 1$. □

4.1 Multiplication and Division Algorithms

Let $M(N)$ be the amount of working storage needed to multiply $a, b \in R_N[x]$ using the algorithm IP_MUL. Similarly let $Q(N)$ be the amount of working storage needed to divide a by b using the algorithm IP_REM. The working storage used in lines 5,11 and 15 of algorithm IP_MUL and lines 6,12 and 16 of algorithm IP_REM is

$$M(N) = 2P(2d_N - 2, R_{N-1}) + \max(M(N-1), Q(N-1)) \text{ and} \tag{1}$$

$$Q(N) = 2P(2d_N - 2, R_{N-1}) + \max(M(N-1), Q(N-1)). \tag{2}$$

Comparing equations (1) and (2) we see that $M(N) = Q(N)$ for any $N \geq 1$. Hence

$$M(N) = 2P(2d_N - 2, R_{N-1}) + M(N-1). \tag{3}$$

Simplifying (3) gives $M(N) = 2S_N - 2N + 2\sum_{i=1}^{N} S_i$. Using Corollary 4 we have

Theorem 6. $M(N) = Q(N) = 2S_N - 2N + 2\sum_{i=1}^{N} S_i < 6S_N$.

Remark 7. When calling the algorithm IP_MUL to compute $c = a \times b$ where $a, b \in R[x]$, we should use a working storage array $W[0 \ldots w_n]$ such that $w_n \geq M(N)$. Since $M(N) < 6S_N$, the working storage must be big enough to store only six coefficients in L_p.

Let $C(N)$ denote the working storage needed for the operation IP_SCAL_MUL. It is easy to show that $C(N) = M(N-1) + P(2d_N - 2, R_{N-1}) < M(N)$.

4.2 Inversion

Let $I(N)$ denote the amount of working storage needed to invert $c \in R_N$. In lines 6, 9, 12, 14, 16, 17, 19 , 22 and 23 of algorithm IP_INV we use the working storage. We have

$$I(N) = 4P(d_N, R_{N-1}) + P(2d_N - 2, R_{N-1}) + \max(I(N-1), M(N-1), Q(N-1)). \quad (4)$$

But we have $M(N-1) = Q(N-1)$, hence

$$I(N) = 4P(d_N, R_{N-1}) + P(2d_N - 2, R_{N-1}) + \max(I(N-1), M(N-1)). \quad (5)$$

Lemma 8. For $N \geq 1$, we have $M(N) < I(N)$.

Proof. (by contradiction) Assume $M(N) \geq I(N)$. Using (5) we have $I(N) = 4P(d_N, R_{N-1}) + P(2d_N - 2, R_{N-1}) + M(N-1)$. On the other hand using (3) we have $M(N) = 2P(2d_N - 2, R_{N-1}) + M(N-1)$. We assumed $I(N) \leq M(N)$ hence we have $4P(d_N, R_{N-1}) + P(2d_N - 2, R_{N-1}) + M(N-1) \leq 2P(2d_N - 2, R_{N-1}) + M(N-1)$ thus $4P(d_N, R_{N-1}) + P(2d_N - 2, R_{N-1}) \leq 2P(2d_N - 2, R_{N-1}) \Rightarrow 6S_N + 3S_{N-1} - 1 \leq 4S_N - 2S_{N-1} - 2$ which is a contradiction. Thus $I(N) > M(N)$. □

Using Equation (4) and Lemma 8 we conclude that $I(N) = 4P(d_N, R_{N-1}) + P(2d_N - 2, R_{N-1}) + I(N-1)$. Simplifying this yields:

Theorem 9. $I(N) = 4\sum_{i=1}^{N} P(d_i, R_{i-1}) + \sum_{i=1}^{N} P(2d_i - 2, R_{i-1}) = 4\sum_{i=1}^{N}(S_i + S_{i-1}) + \sum_{i=1}^{N}(2S_i - S_{i-1} - 1) = 6S_N + 9\sum_{i=1}^{N-1} S_i - N$.

Using Lemma 2 an upper bound for $I(N)$ is $I(N) < 6S_N + 9S_N = 15S_N$.

4.3 GCD Computation

Let $G(N)$ denote the working storage needed to compute the GCD of $a, b \in R_N[x]$. In lines 4,5,7,9,11 and 12 of algorithm IP_GCD we use the working storage. We have $G(N) = \bar{S}_N + \max(I(N), C(N), Q(N))$. Lemma 8 states that $I(N) > M(N) = Q(N) > C(N)$ hence

$$G(N) = \bar{S}_N + I(N) = S_N + S_{N-1} + 6S_N + 9\sum_{i=1}^{N-1} S_i - N = 7S_N + S_{N-1} + 9\sum_{i=1}^{N-1} S_i - N.$$

Since $I(N) < 15S_N$, we have an upper bound on $G(N)$:

Theorem 10. $G(N) = S_N + S_{N-1} + I(N) < S_N + S_{N-1} + 15S_N < 17S_N$.

Remark 11. The constants 6, 15 and 17 appearing in Theorems 6, 9 and 10 respectively, are not the best possible. One can reduce the constant 6 for algorithm IP_MUL if one also uses the space in the output array C for working storage. We did not do this because it complicates the description of the algorithm and yields no significant performance gain.

5 Benchmarks

We have compared our C library with the Magma (see [1]) computer algebra system. The results are reported in Table 1. For our benchmarks we used $p = 3037000453$, two field extensions with minimal polynomials $\bar{m}_1$ and $\bar{m}_2$ of varying degrees d_1 and d_2 but with $d =$

$d_1 \times d_2 = 60$ constant so that we may compare the overhead for varying d_1. We choose three polynomials a, b, g of the same degree d_x in x with coefficients chosen from R at random. The data in the fifth and sixth columns are the times (in CPU seconds) for computing both $f_1 = a \times g$ and $f_2 = b \times g$ using IP_MUL and Magma version 2.15 respectively. Similarly, the data in the seventh and eighth columns are the times for computing both $\mathrm{quo}(f_1, g)$ and $\mathrm{quo}(f_2, g)$ using IP_REM and Magma respectively. Finally the data in the ninth and tenth columns are the times for computing $\gcd(f_1, f_2)$ using IP_GCD and Magma respectively. The data in the column labeled $\#f_i$ is the number of terms in f_1 and f_2.

Table 1: Timings in CPU seconds on an AMD Opteron 254 CPU running at 2.8 GHz

d_1	d_2	d_x	$\#f_i$	IP_MUL	MAG_MUL	IP_REM	MAG_REM	IP_GCD	MAG_GCD
2	30	40	2460	0.124	0.050	0.123	0.09	0.384	2.26
3	20	40	2460	0.108	0.054	0.106	0.11	0.340	2.35
4	15	40	2460	0.106	0.056	0.106	0.10	0.327	2.39
6	10	40	2460	0.106	0.121	0.105	0.14	0.328	5.44
10	6	40	2460	0.100	0.093	0.100	0.37	0.303	7.84
15	4	40	2460	0.097	0.055	0.095	0.17	0.283	3.27
20	3	40	2460	0.092	0.046	0.091	0.14	0.267	2.54
30	2	40	2460	0.087	0.038	0.087	0.10	0.242	1.85
2	30	80	4860	0.477	0.115	0.478	0.27	1.449	9.41
3	20	80	4860	0.407	0.127	0.409	0.27	1.304	9.68
4	15	80	4860	0.404	0.132	0.406	0.28	1.253	9.98
6	10	80	4860	0.398	0.253	0.400	0.35	1.234	22.01
10	6	80	4860	0.380	0.197	0.381	0.86	1.151	31.57
15	4	80	4860	0.365	0.127	0.364	0.40	1.081	13.49
20	3	80	4860	0.353	0.109	0.353	0.33	1.030	10.59
30	2	80	4860	0.336	0.086	0.337	0.26	0.932	7.83

The timings in Table 1 for *in-place* routines show that as the degree d_x doubles from 40 to 80, the time consistently goes up by a factor of 4 indicating that the underlying algorithms are all quadratic in d_x. This is not the case for Magma because Magma is using a sub-quadratic algorithm for multiplication. We describe the algorithm used by Magma ([9]) briefly. To multiply two polynomials $a, b \in L_p[x]$ Magma first multiplies a and b as polynomials in $\mathbb{Z}[x, z_1, \ldots, z_r]$. It then reduces their product modulo the ideal $\langle m_1, \ldots, m_r, p \rangle$. To multiply in $\mathbb{Z}[x, z_1, \ldots, z_r]$, Magma evaluates each variable successively, beginning with z_r then ending with x, at integers $k_r, \ldots, k_1, k_0$ which are powers of the base of the integer representation which are sufficiently large so that that the product of the two polynomials $a(x, z_1, \ldots, z_r) \times b(x, z_1, \ldots, z_r)$ can be recovered from the product of the two (very) large integers $a(k_0, k_1, \ldots, k_r) \times b(k_0, k_1, \ldots, k_r)$. The reason to evaluate at a power of the integer base is so that evaluation and recovery can be done in linear time. In this way polynomial multiplication in $\mathbb{Z}[x, z_r, \ldots, z_1]$ is reduced to a single (very) large integer multiplication which is done using the FFT. This, note, may not be efficient if the polynomials $a(x, z_1, \ldots, z_r)$ and $b(x, z_1, \ldots, z_r)$ are sparse.

Table 1 shows that our in-place GCD algorithm is a factor of 6 to 27 times faster than Magma's GCD algorithm. Since both algorithms use the Euclidean algorithm, this shows that our in-place algorithms for arithmetic in L_p are efficient. This is the gain we sought to achieve. The reader can observe that as d_1 increases, the timings for IP_MUL decrease which shows there is still some overhead for α_1 of low degree.

5.1 Optimizations in the implementation

In modular algorithms, multiplication in $\mathbb{Z}_p$ needs to be coded carefully. This is because hardware integer division (`%p` in C) is much slower than hardware integer multiplication. One can use Peter Montgomery's trick (see [8]) to replace all divisions by p by several cheaper operations for an overall gain of typically a factor of 2. Instead, we use the following scheme which replaces most divisions by p in the multiplication subroutine for $\mathbb{Z}_p[x]$ by at most one subtraction. We use a similar scheme for the division in $\mathbb{Z}_p[x]$. This makes GCD computation in $L_p[x]$ more efficient as well. We observed a gain of a factor of 5 on average for the GCD computations in our benchmarks.

The following C code explains the idea. Suppose we have two polynomials $a, b \in \mathbb{Z}_p[x]$ where $a = \sum_{i=0}^{d_a} a_i x^i$ and $b = \sum_{j=0}^{d_b} b_j x^j$ where $a_i, b_j \in \mathbb{Z}_p$. Suppose the coefficients a_i and b_i are stored in two Arrays A and B indexed from 0 to d_a and 0 to d_b respectively. We assume elements of $\mathbb{Z}_p$ are stored as signed integers and an integer x in the range $-p^2 < x < p^2$ fits in a machine word. The following computes $c = a \times b = \sum_{k=0}^{d_a+d_b} c_k x^k$.

```
M = p*p;
d_c = d_a+d_b;
for( k=0; k<=d_c; k++ ) {
   t = 0;
   for( i=max(0,k-d_b); i <= min(k,d_a); i++ )
   {
      if( t<0 ); else t = t-M;
      t = t+A[i]*B[k-i];
   }
   t = t % p;
   if( t<0 ) t = t+p;
   C[k] = t;
}
```

The trick here is to put t in the range $-p^2 < t \leq 0$ by subtracting p^2 from it when it is positive so that we can add the product of two integers $0 \leq a_i, b_{k-i} < p$ to t without overflow. Thus the number of divisions by p is linear in d_c, the degree of the product. One can further reduce the number of divisions by p. In our implementation, when multiplying elements $a, b \in \mathbb{Z}_p[z][x]/\langle m(z) \rangle$ we multiply $a, b \in \mathbb{Z}_p[z][x]$ without division by p before dividing by $m(z)$.

Note that the statement `if( t<0 ); else t = t-M;` is done this way rather than the more obvious `if( t>0 ) t = t-M;` because it is faster. The reason is that $t < 0$ holds about 75% of the time and the code generated by the newer compilers is optimized for the case the condition of an if statement is true. If one codes the if statement using `if( t>0 ) t = t-M;` instead, we observe a loss of a factor of 2.6 on an Intel Core i7, 2.3 on an Intel Core 2 duo, and 2.2 on an AMD Opteron for the above code.

6 Concluding Remarks

Our C library of in-place routines has been integrated into Maple 14 for use in the GCD algorithms in [11] and [4]. These algorithms compute GCDs of polynomials in $K[x_1, x_2, \ldots, x_n]$ over an algebraic function field K in parameters $t_1, t_2, \ldots, t_k$ by evaluating the parameters and variables except x_1 and using rational function interpolation to recover the GCD. This

results in many GCD computations in $L_p[x_1]$. In many applications, K has field extensions of low degree, often quadratic or cubic.

Our C library is available on our website at
`http://www.cecm.sfu.ca/CAG/code/ASCM09/inplace.c`
The code we used to generate the Magma timings in Section 5 is available in the file
`http://www.cecm.sfu.ca/CAG/code/ASCM09/magma.txt`

In [6], Xin, Moreno Maza and Schost develop asymptotically fast algorithms for multiplication in L_p based on the FFT and use their algorithms to implement the Euclidean algorithm in $L_p[x]$ for comparison with Magma and Maple. The authors obtain a speedup for L of sufficiently large degree d. Our results in this paper are complementary in that we sought to improve arithmetic when L has relatively low degree.

Acknowledgments

This work was supported by the MITACS NCE of Canada.

References and Notes

[1] Wieb Bosma, John J. Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, **24**(3–4):235–266, 1997.

[2] Mark J. Encarnación. Computing gcds of polynomials over algebraic number fields. *J. Symb. Comp.*, **20**(3):299–313, 1995.

[3] Richard Fateman. Comparing the speed of programs for sparse polynomial multiplication. *SIGSAM Bulletin*, **37**(1):4–15, ACM Press, 2003.

[4] S. M. M. Javadi and M. B. Monagan. A sparse modular gcd algorithm for polynomials over algebraic function fields. *Proceedings of ISSAC '07*, pp. 187–194, ACM Press, 2007.

[5] Lars Langemyr and Scott McCallum. The computation of polynomial greatest common divisors over an algebraic number field. *J. Symbolic Comput.*, **8**(5):429–448, 1989.

[6] Xin Li, Marc Moreno Maza, and Éric Schost. Fast arithmetic for triangular sets: from theory to practice. *Proceedings of ISSAC '07*, pp. 269–276, ACM Press, 2007.

[7] Michael B. Monagan. In-place arithmetic for polynomials over $\mathbb{Z}_n$. *Proceedings of DISCO '92*, pp. 22–34, Springer-Verlag, 1993.

[8] Peter Montgomery. Modular multiplication without trial division. *Math. Comp.*, **44**(70):519–521, 1985.

[9] Allan Steel. Multiplication in $L_p[x]$ in Magma. Private communication, 2009.

[10] Mark van Hoeij and Michael Monagan. A modular gcd algorithm over number fields presented with multiple extensions. *Proceedings of ISSAC '02*, ACM Press, pp.109–116, 2002.

[11] Mark van Hoeij and Michael Monagan. Algorithms for polynomial gcd computation over algebraic function fields. *Proceedings of ISSAC '04*, ACM Press, pp. 297–304, 2004.

MACIS2009

Polynomial system solving
Systems and Control
Software Science

Intersection Formulas and Algorithms for Computing Triangular Decompositions

Changbo Chen and Marc Moreno Maza

[1] ORCCA, University of Western Ontario (UWO)
London, Ontario, Canada
{cchen252,moreno}@csd.uwo.ca

Abstract

Methods for computing triangular decompositions of polynomial systems can be classified into two groups. First, those computing a series of regular chains $C_1 \ldots, C_e$ such that for each irreducible component V of the variety of the input system, one of the C_i's encodes a generic zero of V. An example is the algorithm of Michael Kalkbrener in his PhD thesis. Secondly are those methods computing a series of characteristic sets $C_1 \ldots, C_f$ (in the sense of Wu Wen Tsün) such that the variety of the input system is the union of the quasi-components of the C_i's.
A large variety of methods fall in the second family, in particular the one proposed in 1987 by Wu Wen Tsün in the 1st volume of the *MM Research Preprints*. Some methods belong to both families, this is the case for those proceeding in an incremental manner, that is, solving one equation after another. These latter methods rely on an operation for computing the intersection of an hypersurface and the quasi-component of a regular chain. This is an attractive operation since its input can be regarded as well-behaved geometrical objects. However, known algorithms (the one of Daniel Lazard in 1991 and the one of the second author in 2000) are quite involved and difficult to analyze. We revisit this intersection operation. We show that under "genericity assumptions" a simple algorithm can be stated and analyzed. In this context, it follows a classical *projection-extension* scheme. However, we shall see that the cost of the extension step can be neglected in comparison to that of the projection, which itself "essentially" reduces to a cost that all such intersection algorithms have to pay. In our experimental results, realized with the RegularChains library in Maple, our new intersection outperforms the one of the second author by several orders of magnitude on sufficiently difficult problems.

Root Isolation of Zero-dimensional Polynomial Systems with Linear Univariate Representation(Abstract) *

Jin-San Cheng, Xiao-Shan Gao, Leilei Guo
Key Lab of Mathematics Mechanization
Institute of Systems Science, Academia Sinica, Beijing, China
xgao@mmrc.iss.ac.cn,jcheng@amss.ac.cn

In this paper, we present a new explicit representation for the roots of a zero-dimensional polynomial system: the roots are represented as linear combinations of some roots of several univariate polynomial equations, which is a generalization of the method proposed in [1]. We call this representation as **Linear Univariate Representation** (shortly LUR). This gives an algebraic representation for the roots of the system. It is different from the well-known RUR representation which is based on the idea of generic position [3]. From LUR, we can obtain the roots of the system under any given precision by refining the isolation intervals of these univariate polynomial equations at most once.

1 Linear univariate representation

Let

$$\mathcal{P} = \{f_1(x_1, \ldots, x_n), \ldots, f_s(x_1, \ldots, x_n)\}$$

be a zero-dimensional polynomial system in $\mathbb{Q}[x_1, \ldots, x_n]$, where $\mathbb{Q}$ is the field of rational numbers. We use $V_{\mathbb{C}}(\mathcal{P})$ to denote its roots in $\mathbb{C}^n$, where $\mathbb{C}$ is the field of complex numbers. Denote $(\mathcal{P})$ to be the ideal generated by $\mathcal{P}$ and

$$\mathcal{I}_i = (\mathcal{P}) \cap \mathbb{Q}[x_1, \ldots, x_i], i = 1, \ldots, n.$$

By a linear univariate representation (abbr. LUR), we mean a set like

$$\{T_1(x), \ldots, T_n(x), s_i, d_i, i = 1, \ldots, n-1\} \tag{1}$$

where $T_i \in \mathbb{Q}[x]$ are univariate polynomials, s_i and d_i are positive rational numbers. The roots of (1) are defined to be

$$\begin{aligned} &\{(\alpha_1, \frac{\alpha_2 - \alpha_1}{s_1}, \ldots, \frac{\alpha_n - \alpha_{n-1}}{s_1 \cdots s_{n-1}}) \,|\, T_i(\alpha_i) = 0, i = 1, \ldots, n \text{ and} \\ &|\alpha_{i+1} - \alpha_i| < s_0 \cdots s_{i-1} d_i, i = 1, \ldots, n-1, \text{ where } s_0 = 1.\} \end{aligned} \tag{2}$$

An LUR for $\mathcal{P} = 0$ is a set of form (1) whose roots are exactly the roots of $\mathcal{P} = 0$.

*Partially supported by a National Key Basic Research Project of China and by a USA NSF grant CCR-0201253.

It is clear that an LUR represents the roots of $\mathcal{P}$ as linear combinations of the roots of univariate polynomial equations. The LUR representation has the following advantage: we can easily derive the precision of the roots of $\mathcal{P}$ from that of the univariate equations.

Lemma 1.1 *If α_i is a root of $T_i(x) = 0$ with precision ϵ_i, then the root $(\alpha_1, \frac{\alpha_2-\alpha_1}{s_1}, \ldots, \frac{\alpha_n-\alpha_{n-1}}{s_1\cdots s_{n-1}})$ of $\mathcal{P}$ has precision $\max\{\epsilon_1, \frac{\epsilon_2+\epsilon_1}{s_1}, \ldots, \frac{\epsilon_n+\epsilon_{n-1}}{s_1\cdots s_{n-1}}\}$.*

For a zero-dimensional polynomial system $\mathcal{P}$, let $d_i, r_{i+1}, s_i, i = 1, \ldots, n-1$ be rational numbers satisfying

$$\begin{aligned} d_i &< \min\{\frac{1}{2}|\alpha-\beta|, \forall \eta \in V_{\mathbb{C}}(\mathcal{I}_{i-1}), (\eta,\alpha), (\eta,\beta) \in V_{\mathbb{C}}(\mathcal{I}_i), \alpha \neq \beta\}, \\ r_i &> 2\max\{|\alpha_i|, \forall(\alpha_1,\ldots,\alpha_i) \in V_{\mathbb{C}}(\mathcal{I}_i)\}, \\ 0 &< s_i < \frac{d_i}{r_{i+1}}. \end{aligned} \tag{3}$$

Geometrically, d_i (r_i) is half (double) of the separation bound (root bound) for roots of $\mathcal{I}_i$ considered as points on a "fiber" over each root of $\mathcal{I}_{i-1}$, and s_i, meaning the slope of certain line, is a key parameter to be used in our method. Note that $d_i = +\infty$ if $\forall \eta \in V_{\mathbb{C}}(\mathcal{I}_{i-1})$, $\#\{\alpha|(\eta,\alpha) \in V_{\mathbb{C}}(\mathcal{I}_i)\} = 1$. We can choose any positive number as d_i.

For s_i satisfying (3), consider the ideal

$$\bar{\mathcal{I}}_i = (\mathcal{I}_i \cup \{x - x_1 - s_1x_2 - \cdots - s_1\cdots s_{i-1}x_i\})$$

where x is a new variable. It is clear that $\bar{\mathcal{I}}_i$ is a zero-dimensional ideal in $\mathbb{Q}[x_1,\ldots,x_i,x]$. And the elimination ideal $(\bar{\mathcal{I}}_i) \cap \mathbb{Q}[x]$ is principal. Let $T_i(x)$ be the generator of this ideal:

$$(\bar{\mathcal{I}}_i) \cap \mathbb{Q}[x] = (T_i(x)). \tag{4}$$

Theorem 1.2 *If s_i, d_i satisfy condition (3) and T_i is defined in (4), then the corresponding set (1) is an LUR for $\mathcal{P}$.*

2 Algorithm for computing an LUR and roots isolation

We use intervals to isolate real numbers: let $\square\mathbb{Q}$ denote the set of intervals of the form $[a,b]$ where $a \leq b \in \mathbb{Q}$. The **length** of an interval $I = [a,b] \in \square\mathbb{Q}$ is defined to be $|I| = b - a$.

Let $\langle\alpha,\beta\rangle = \alpha + \beta\mathfrak{i}$ ($\mathfrak{i}^2 = -1$) be a complex root of $T(x) = 0$. We call $\langle I^{\mathbb{R}}, I^{\mathbb{I}}\rangle$ ($I^{\mathbb{R}}, I^{\mathbb{I}} \in \square\mathbb{Q}$) its **isolation interval** if it contains only one root $\langle\alpha,\beta\rangle$. The length of the isolation interval of $\langle\alpha,\beta\rangle$ is $\max\{|I^{\mathbb{R}}|, |I^{\mathbb{I}}|\}$. We call $\mathbf{B}^{\mathbb{C}} = \langle I_1^{\mathbb{R}}, I_1^{\mathbb{I}}\rangle \times \cdots \times \langle I_n^{\mathbb{R}}, I_n^{\mathbb{I}}\rangle$ an **ϵ-isolation box** of a root of $\mathcal{P}$ if $\mathbf{B}^{\mathbb{C}}$ contains only one root of $\mathcal{P}$ and $\max_i\{|I_i^{\mathbb{R}}|, |I_i^{\mathbb{I}}|\} \leq \epsilon$.

The main step of the algorithm is as follows.

1. We compute a Gröbner basis $\mathcal{G}$ of $\mathcal{P}$ under the pure lexicographical order induced by $x_1 < \cdots < x_n$ with the method introduced in [2]. The advantage of this approach is that its worst case complexity is of single exponential. It is clear that $\mathcal{G}$ is of the form:

$$\mathcal{G} = \begin{cases} P_1(x_1) \\ P_{2,1}(x_1,x_2)\ldots, P_{2,k_2}(x_1,x_2) \\ \cdots \\ P_{n,1}(x_1,\ldots,x_n),\ldots,P_{p,k_n}(x_1,\ldots,x_n) \end{cases} \tag{5}$$

where $P_{i,j} \in \mathbb{Q}[x_1, \ldots, x_i]$. Let $\mathcal{G}_i = \{P_1, P_{2,1}, \ldots, P_{2,k_2}, \ldots, P_{i,1}, \ldots, P_{i,k_i}\}$.

2. Compute a linear univariate representation (LUR) of $\mathcal{P} = 0$ and roots isolation of $\mathcal{P} = 0$ from the Gröbner basis (5) of $\mathcal{P}$.

We will compute the LUR (1) recursively. Assume that we have computed an LUR of $\mathcal{G}_i$, the ϵ-isolation boxes of the roots of $\mathcal{G}_i$ $(1 \leq i \leq n-1)$ and also d_i. It is reasonable since $T_1(x) = P_1(x)$ and we can isolate the roots of $T_1(x) = 0$ and derive d_1. We will show how to compute an LUR for $\mathcal{G}_{i+1}$ and ϵ-isolation boxes of the roots of $\mathcal{G}_{i+1} = 0$. We will compute r_{i+1}, s_i as introduced in (3), then compute $T_{i+1}(x)$ defined in (4) for $1 \leq i \leq n-1$.

Now we show how to estimate r_{i+1}. With the method in [2], we compute a univariate polynomial in x_{i+1}: $(g_{i+1}(x_{i+1})) = (\mathcal{G}_{i+1}) \cap \mathbb{Q}[x_{i+1}]$.

Lemma 2.1 *Use the notations introduced before. Then we can take*

$$r_{i+1} = 2\max\{\mathrm{RB}(g_{i+1}(x_{i+1}))\}, \tag{6}$$

where $\mathrm{RB}(g)$ *is the root bound of a univariate polynomial equation* $g = 0$.

Then we can choose a rational number s_i such that the third formula of (3) holds. We need also to choose s_i to ensure that there is no polynomial $f \in \mathcal{G}_{i+1} - \mathcal{G}_i$ such that $h_{i+1} = x - x_1 - s_1 x_2 - \ldots - s_1 \cdots s_i\, x_{i+1}\, mod\, f \in \mathbb{Q}[x]$. Otherwise, the hyperplane $h_{i+1} = 0$ is parallel to the hyperplane $f = 0$, then $T_{i+1}(x)$ is of degree 1.

We will compute $T_{i+1}(x)$. Note that

$$\bar{\mathcal{G}}_{i+1} = \mathcal{G}_{i+1} \cup \{x - x_1 - s_1 x_2 - \ldots - s_1 \cdots s_i\, x_{i+1}\}$$

is still a Gröbner basis under the variable order $x_1 < \ldots < x_i < x$. We can compute $T_{i+1}(x)$ such that $(\bar{\mathcal{G}}_{i+1}) \cap \mathbb{Q}[x] = (T_{i+1}(x))$ with the method in [2].

After $T_{i+1}(x)$ is obtained, we derive an LUR for $\mathcal{G}_{i+1}$. We will compute the ϵ-isolation boxes for the roots of $\mathcal{G}_{i+1} = 0$ as follows.

Consider $x - x_1 - s_1\, x_2 - \ldots - s_1 \ldots s_i x_{i+1} = 0$ in $\bar{\mathcal{G}}_{i+1}$. It is clear that $x' = x_1 + s_1\, x_2 + \ldots + s_1 \ldots s_{i-1} x_i$ is a root of $T_i(x) = 0$ when $(x_1, \ldots, x_i) \in V_{\mathbb{C}}(\mathcal{G}_i)$. We have

$$x_{i+1} = \frac{x - x'}{s_1 \cdots s_i}, \tag{7}$$

$$|x - x'| = |s_1 \cdots s_i x_{i+1}| < s_1 \cdots s_i (r_{i+1} - \mathrm{RB}(g_{i+1})) < s_1 \cdots s_{i-1} d_i - s_1 \cdots s_i d_i/2. \tag{8}$$

From the formula above and the way we compute d_i, the isolation intervals of the roots of $T_{i+1}(x) = 0$ corresponding to the root $(\xi_1, \ldots, \xi_i) \in V_{\mathbb{C}}(\mathcal{G}_i)$ is exactly inside

$$\mathbb{I}_{\eta_i} = \langle (Re(\eta_i) - s_1 \cdots s_{i-1} d_i, Re(\eta_i) + s_1 \cdots s_{i-1} d_i), (Im(\eta_i) - s_1 \cdots s_{i-1} d_i, Im(\eta_i) + s_1 \cdots s_{i-1} d_i) \rangle$$

if we isolate the roots of $T_{i+1}(x) = 0$ with precision no larger than $s_1 \cdots s_i d_i/2$, where $\eta_i = Re(\eta_i) + Im(\eta_i)\mathfrak{i}$ is the corresponding root of $(\xi_1, ..., \xi_i)$ in $T_i(x_i) = 0$ and $\langle [a, b], [c, d] \rangle$ is its isolation interval.

We will compute the isolation intervals of the roots of $\mathcal{G}_{i+1} = 0$ at $(\eta_1, \ldots, \eta_i) \in V_{\mathbb{C}}(\mathcal{G}_i)$ to ensure that they are disjoint and their lengths are no larger than ϵ.

Assume $a_1 \leq a_2, p_1 \leq p_2$, we define the distance between two isolation intervals:

$$Dis([a_1, b_1], [a_2, b_2]) = \begin{cases} a_2 - b_1, & \text{if } [a_1, b_1] \cap [a_2, b_2] = \emptyset, \\ 0, & \text{otherwise,} \end{cases}$$

$Dis(\langle[a_1,b_1],[p_1,q_1]\rangle,\langle[a_2,b_2],[p_2,q_2]\rangle) = \max\{Dis([a_1,b_1],[a_2,b_2]), Dis([p_1,q_1],[p_2,q_2]\}$.

Let $\langle[p_j,q_j],[g_j,h_j]\rangle (1 \le j \le m)$ be the isolation intervals for the roots of $T_{i+1}(x)=0$ in $\mathbb{I}_{\eta_i}$. Then from (7), the isolation intervals of $\eta_{i+1,j} (1 \le j \le m)$ is

$$J_{i+1,j} = \frac{\langle[p_j,q_j]-[a,b],[g_j,h_j]-[c,d]\rangle}{s_1\cdots s_i} = \frac{\langle[p_j-b,q_j-a],[g_j-d,h_j-c]\rangle}{s_1\cdots s_i} \tag{9}$$

We have two conditions (One is to ensure the precision (10), the other is to ensure any two insolation interval are disjoint (11).) to derive ϵ-isolation boxes ($1 \le j \le m$):

$$(q_j-p_j)+(b-a) < s_1\cdots s_i\epsilon,\ \ (g_j-h_j)+(d-c) < s_1\cdots s_i\epsilon \tag{10}$$

$$D_{\eta,i+1} = \min_{1\le k\ne j\le m} Dis(\langle[p_k,q_k],[g_k,h_k]\rangle,\langle[p_j,q_j],[g_j,h_j]\rangle) > \max\{b-a,d-c\}. \tag{11}$$

We also derive

$$d_{i+1} = \min_{\eta\in V_{\mathbb{C}}(\mathcal{G}_i)}\{\frac{d_i}{2s_i}, \frac{D_{\eta,i+1}-\max\{b-a,d-c\}}{2s_1\cdots s_i}\}. \tag{12}$$

In the end, we obtain the ϵ-isolation boxes for $\mathcal{P}=0$ and also an LUR for $\mathcal{P}$.

We can write the above result as a theorem.

Theorem 2.2 *Using the same notations as before. Let ϵ_i be the precision to isolate the roots of $T_i(x)=0$, and θ_{i+1} is the minimum of the distance of any two isolation intervals of $T_{i+1}(x)=0$ $(1\le j\le n-1)$. If*

$$\epsilon_i+\epsilon_{i+1} < s_1\cdots s_i\epsilon,\ \ \epsilon_{i+1} < \frac{s_1\cdots s_i\, d_i}{2},\ \ \epsilon_i < \min\{\theta_{i+1}, \frac{d_i}{2\, s_i}\},$$

and we construct the last isolation interval of the isolation boxes of the roots of $\mathcal{G}_{i+1} (1 \le j \le n-1)$ by (9), the derived isolation boxes for $\mathcal{G}_n=0$ are ϵ-isolation boxes.

Example 2.3 $\mathcal{P} := [x^2+y^2+z^2-3, x^2+2*y^2-3*z+1, x+y-z]$.
An LUR of $\mathcal{P}$ is as follows: $[[T_1(t),T_2(t),T_3(t)],[s_1,s_2],[d_1,d_2]] = [[5-60\,t+6\,t^2+18\,t^3+6\,t^4, 863337-6119640\,t+360000\,t^2+1920000\,t^3+640000\,t^4, 53294617-309903360\,t+11884800\,t^2+94464000\,t^3+30720000\,t^4], [1/20,1/2],[1/2,5]]$. *It has four complex roots. The roots of $\mathcal{P}$ are:* $[(\alpha, 20(\beta-\alpha), 40(\gamma-\beta))|T_1(\alpha)=0, T_2(\beta)=0, T_3(\gamma)=0, |\beta-\alpha|<1/2, |\gamma-\beta|<1/4]$. *Assuming the final precision $\epsilon = 1/2^{10}$ for the real roots of the system, we can set $\epsilon_1=\frac{1}{40}\epsilon, \epsilon_2=\epsilon_3=\frac{1}{80}\epsilon$. Its two real roots with the given precision are*
$[\frac{5519}{65536},\frac{345}{4096}]\times[\frac{4835}{4096},\frac{38695}{32768}]\times[\frac{20715}{16384},\frac{20725}{16384}],\ [\frac{44479}{32768},\frac{88959}{65536}]\times[\frac{-10985}{32768},\frac{-5485}{16384}]\times[\frac{16745}{16384},\frac{16755}{16384}]$.

References and Notes

[1] J.S. Cheng, X.S. Gao, J. Li, Root isolation for bivariate polynomial systems with local generic position method. *Proc. ISSAC 2009*, 103-109, ACM Press, New York, 2009.

[2] J.C. Faugere, P. Gianni, D. Lazard, T. Mora, Efficient computation of zero-dimensional Gröbner bases by change of ordering, *Journal of Symbolic Computation*, 16(4), 329 - 344, 1993.

[3] F. Rouillier, Solving zero-dimensional systems through the rational univariate representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5), 433–461, 1999.

Efficient computation of square-free Lagrange resolvents

Antoine COLIN, Marc GIUSTI*

LIX, UMR CNRS-Polytechnique 7161,
École polytechnique, F-91128 Palaiseau Cedex France
ph. 33 - 1 69 33 40 83
acolin@univ-lr.fr, giusti@lix.polytechnique.fr

Abstract

If H is a finite subgroup of the general linear group $\mathbf{GL}_n(k)$, we propose a general frame to compute efficiently in the invariant algebra $k[X_1,\ldots,X_n]^H$. The classical Noether normalization of this Cohen-Macaulay algebra takes a natural form when expressed with adequate data structures, based on evaluation rather than writing. This allows to compute more efficiently its multiplication tensor.
As an illustration we give a fast symbolic algorithm to compute the coefficients of the Lagrange resolvent associated to the given subgroup H, either generically or specialized. We show also how to find square-free resolvents with better theoretical complexity (polynomial in the index of the group after a precomputation depending only on H). This relies on a geometric link between the discriminant of the natural Noether projection and two other discriminants related to fundamental invariants.

A complete text with the bibliography can be found at
http://lix.polytechnique.fr/~giusti

1 Introduction and result

Let k be a field of characteristic 0. In all that follows, H is a finite subgroup of the general linear group $\mathbf{GL}_n(k)$. We consider the right action of the group $\mathbf{GL}_n(k)$ on the polynomial ring $k[\mathbf{X}] = k[X_1,\ldots,X_n]$, defined by the following action of the matrix $A = (a_{i,j})$ on the polynomial p:

$$(p, A) \mapsto p^A = p(A.\mathbf{X}) = p\left(a_{11}X_1 + \ldots a_{1n}X_n, \ldots, a_{n1}X_1 + \cdots + a_{nn}X_n\right)$$

The invariant polynomials under this action form the invariant algebra denoted by $k[\mathbf{X}]^H$, equipped with the induced graded structure inherited from $k[\mathbf{X}]$.

The general linear group has also a left action on the affine space $\mathbb{A}^n_k \simeq k^n$: for any point $\boldsymbol{x} \in k^n$, $A.\boldsymbol{x}$ is defined as the usual product of the matrix A and the column $\boldsymbol{x}$. This left action is coherent with the right action on $k[\mathbf{X}]$, in the sense that $p^A(\boldsymbol{x}) = p(A.\boldsymbol{x})$.

We consider the symmetric group $\mathfrak{S}_n$ as a subgroup of $\mathbf{GL}_n(k)$ by identifying a permutation τ with the *permutation matrix* $A_\tau = (\delta_{i,\tau(j)})$. It induces a right action of $\mathfrak{S}_n$ on $k[\mathbf{X}]$. Therefore, in the following, the case $H \subset \mathfrak{S}_n$ will be considered as a subcase of $H \subset \mathbf{GL}_n(k)$.

In this framework, Hochster-Eagon's theorem states that the invariant algebra $k[\mathbf{X}]^H$ is Cohen-Macaulay, and, using the Noether normalization lemma, admits a natural **Hironaka decomposition** in terms of **primary** invariants $\mathbf{\Pi} = (\Pi_1,\ldots,\Pi_n)$ and **secondary**

*Correspondence to: LIX, Polytechnique, F-91128 Palaiseau Cedex France, +33 1 69 33 40 83.

invariants $\mathbf{\Sigma} = (\Sigma_1, \ldots, \Sigma_r)$:

$$k[\mathbf{X}]^H = \bigoplus_{i=1}^{r} k[\mathbf{\Pi}]\Sigma_i \qquad \text{(direct sum of } k[\mathbf{\Pi}]\text{-modules)}$$

where the Π_i are algebraically independent over k. In the case of a permutation subgroup H of $\mathfrak{S}_n$, the **elementary symmetric polynomials** are always a possible choice. The Σ_i are algebraic integers and are linearly independent over $k[\mathbf{\Pi}]$. The inclusion $k[\mathbf{\Pi}] \hookrightarrow k[\mathbf{X}]^H$ realizes the integral extension of a Noether normalization. The number r of secondary invariants is then $[\mathfrak{S}_n : H]$.

On the other hand, M. Giusti, J. Heintz, L. M. Pardo and their collaborators showed in a sequence of papers that a Noether position is a good frame for fast computations in the context of multivariate polynomial algebras. The reason is that it enables to use an adequate data structure (straight-line programs) to store with better complexity the free (or transcendental) variables. In particular this explains why fast evaluation techniques work when specializing these variables.

In this paper, we show that this idea has a new application in computational geometric invariant theory: considering primary invariants as free variables will allow to compute more efficiently in invariant algebras under finite groups. As an illustration we obtain:

Theorem *There exists an algorithm that computes a square-free Lagrange H-resolvent of a univariate polynomial in polynomial time in the index of the group H, after a precomputation depending only on H.*

2 Geometry of the problem

The geometric properties considered in this section and the following need to assume that the ground field k is **algebraically closed**.

Usually in computer algebra, a variety is naturally embedded in a given ambient space since it is defined by equations. So its algebra of functions is a quotient of a regular algebra. On the opposite here, the invariant algebra $k[\mathbf{X}]^H$ is a subalgebra of a regular algebra. From the section above, it can be seen as the algebra of functions on the algebraic variety $\mathcal{V} = \mathbf{V}(\mathfrak{I}) \subset \mathbb{A}^{n+r}$, which is irreducible ($k[\mathbf{X}]^H$ is a domain, so $\mathfrak{I}$ is prime).

Call respectively $\boldsymbol{x}$, $\mathbf{\Pi}(\boldsymbol{x})$ and $\mathbf{\Sigma}(\boldsymbol{x})$ the points:
$(x_1, \ldots, x_n)$, $(\Pi_1(\boldsymbol{x}), \ldots, \Pi_n(\boldsymbol{x}))$, $(\Sigma_1(\boldsymbol{x}), \ldots, \Sigma_r(\boldsymbol{x}))$. Let

$$\varphi : \left| \begin{array}{ccc} \mathbb{A}_k^n & \longrightarrow & \mathcal{V} \\ \boldsymbol{x} & \longmapsto & \varphi(\boldsymbol{x}) = (\mathbf{\Pi}(\boldsymbol{x}), \mathbf{\Sigma}(\boldsymbol{x})) \end{array} \right.$$

We have two notions of quotient. First, the **categorical quotient** $\mathbb{A}_k^n /\!\!/ H$, defined as the affine variety corresponding to the ring $k[\mathbf{X}]^H$. The projection φ realizes the embedding $\mathcal{V}$ in $\mathbb{A}_k^{n+r}$ of this categorical quotient.

Second, the classical set quotient $\mathbb{A}_k^n / H$, defined as the set of orbits under H, associated to the orbit projection

$$\left| \begin{array}{ccc} \mathbb{A}_k^n & \longrightarrow & \mathbb{A}_k^n / H \\ \boldsymbol{x} & \longmapsto & H.\boldsymbol{x} \end{array} \right. .$$

The categorical quotient $\mathcal{V}$ is the image of φ, and coincides with the quotient set $\mathbb{A}_k^n/H$. We say that the categorical quotient is a **geometric quotient**: the quotient map of the projection φ by the orbit projection realizes a bijection from $\mathbb{A}_k^n/H$ onto $\mathcal{V}$, the embedding of the affine variety $\mathbb{A}_k^n/\!\!/H$.

The projection $p : \left| \begin{array}{ccc} \mathcal{V} & \longrightarrow & \mathbb{A}_k^n \\ (\boldsymbol{\pi}, \boldsymbol{\sigma}) & \longmapsto & \boldsymbol{\pi} \end{array} \right.$ achieves a Noether position w.r.t. the free variables $\mathbf{\Pi}$ is called the **Noether projection**.

We call **primary projection** the map defined from the primary invariants:

$\varpi : \left| \begin{array}{ccc} \mathbb{A}_k^n & \longrightarrow & \mathbb{A}_k^n \\ \boldsymbol{x} & \longmapsto & \varpi(\boldsymbol{x}) = \mathbf{\Pi}(\boldsymbol{x}) \end{array} \right.$.

We get the two following commutative diagrams, where ψ is the canonical injection.

$$\begin{array}{ccccc} \mathbb{A}_k^n \times \mathbb{A}_k^r & \xleftarrow{\ \psi\ } & \mathcal{V} & \xleftarrow{\ \varphi\ } & \mathbb{A}_k^n \\ & {\scriptstyle pr_1}\searrow & \downarrow{\scriptstyle p} & \swarrow{\scriptstyle \varpi} & \\ & & \mathbb{A}_k^n & & \end{array} \qquad (D_1)$$

All the maps in the right triangle of this diagram are finite hence proper (indeed, $k[\mathbf{X}]$ is integral over $k[\mathbf{X}]^H$, which is itself integral over $k[\mathbf{\Pi}]$).

$$\begin{array}{ccccccccc} 0 & \longrightarrow & \mathfrak{I} & \longrightarrow & k[\mathbf{Y},\mathbf{Z}] & \longrightarrow & k[\mathbf{Y},\mathbf{Z}]/\mathfrak{I} & \longrightarrow & k[\mathbf{X}] \\ & & & & || & & || & & || \\ & & & & \mathcal{O}_{\mathbb{A}_k^{n+r}} & \xrightarrow{\psi^* = \Psi} & \mathcal{O}_{\mathcal{V}} & \xrightarrow{\varphi^*} & \mathcal{O}_{\mathbb{A}_k^n} \\ & & & & & {\scriptstyle pr_1^*}\nwarrow & \uparrow{\scriptstyle p^*} & \nearrow{\scriptstyle \varpi^*} & \\ & & & & & & k[\mathbf{Y}] & & \end{array} \qquad (D_2)$$

An important feature of this geometric presentation relies in the relationship between the discriminants of the three maps φ, p and $\varpi = p \circ \varphi$: $\mathcal{D}(\varpi) = \mathcal{D}(p) \cup p(\mathcal{D}(\varphi))$ (where $\mathcal{D}$ denotes the discriminant).

Let us introduce r new variables $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_r)$ over k^r. The multiplication by $\Theta_{\boldsymbol{\lambda}} = \lambda_1\Sigma_1 + \cdots + \lambda_r\Sigma_r$ is an endomorphism of the free $k[\mathbf{\Pi}]$-module $k[\mathbf{X}]^H$. Its characteristic polynomial is the generic Lagrange resolvent $\mathcal{L}_{\Theta_{\boldsymbol{\lambda}}}$ of $\Theta_{\boldsymbol{\lambda}}$. It can be computed from the multiplication table between the elements of the basis $\mathbf{\Sigma}$ of the $k[\mathbf{\Pi}]$-module $k[\mathbf{X}]^H$.

We suppose this multiplication table precomputed once and for all.

3 Application to Lagrange resolvents

The geometric framework developed above finds an application to the computation of Lagrange resolvents. One of the interests of these lies in the direct Galois problem, i.e. describing the Galois group of a given polynomial. Observe however that this problem is not our main concern here. To illustrate this point of view, we give at the end an enlightening example which has its own interest.

From now on, let H be a subgroup of $\mathfrak{S}_n$ and let us take the elementary symmetric polynomials as primary invariants. In this situation the discriminant of the Noether projection and the (irreducible) discriminant of the primary projection coincide.

Let now $f(T) = T^n - \pi_1 T^{n-1} + \cdots + (-1)^n \pi_n$ be a univariate polynomial of degree n. We specialize the $k[\mathbf{\Pi}]$-algebra structure on $k[\mathbf{X}]^H$ into a k-algebra structure, by specializing Π_i in π_i in the precomputed multiplication table between the elements of the basis $\mathbf{\Sigma}$. The Lagrange resolvent $\mathcal{L}_{\Theta_{\boldsymbol{\lambda}},f}$ of f is the specialisation of the generic one at $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$. We compute it thanks to the specialized multiplication table. This algorithm to compute $\mathcal{L}_{\Theta_{\boldsymbol{\lambda}},f}$ is in fact a straigt line program defining the evaluation of $\mathcal{L}_{\Theta_{\boldsymbol{\lambda}}}$. This evaluation algorithm is even quicker than evaluating the precomputed generic $\mathcal{L}_{\Theta_{\boldsymbol{\lambda}}}$.

If $\boldsymbol{\pi}$ is outside the discriminant of the Noether projection (equal to the discriminant of the generic monic univariate polynomial of degree n, hence irreducible), then we can find a point $\boldsymbol{\lambda}$ such that $\mathcal{L}_{\Theta_{\boldsymbol{\lambda}},f}$ is squarefree. For such fixed $\boldsymbol{\pi}$ and $\boldsymbol{\lambda}$, we call $\Theta_{\boldsymbol{\lambda}}$ a *separating* or *primitive* invariant.

The bad choices of the $\boldsymbol{\lambda}$ are enclosed in a hypersurface. As we know its degree and a good upper bound on the evaluation complexity of its equation, we can find a good choice with a good complexity thanks to the Heintz–Schnorr theorem.

Let us finish with the example of the metacyclic subgroup H of index 6 of $\mathfrak{S}_5$, generated by the permutations $((1,2,3,4,5),(2,3,5,4))$. The secondary invariants can be taken as follows: $\Sigma_1 = 1$, Σ_2 is the sum of the monomials of the orbit of $X_1^2 X_2 X_3$, Σ_3 is the sum of the monomials of the orbit of $X_1^3 X_2 X_3$, ...

Let us consider the general polynomial of degree 5 under its Bring–Jerrard form:

$$f(T) = T^5 + \pi_4 T - \pi_5$$

and consider only elements $\Theta_{\boldsymbol{\lambda}}$ of the form $\lambda_2\Sigma_2 + \lambda_3\Sigma_3$.

The bad hypersurface has equation:

$$\delta(\pi_4, \pi_5)^3 (\lambda_3 \,\square + 976565\, \lambda_2^{15}\, {\pi_5}^6)^2 = 0$$

where $\delta(\pi_4, \pi_5)$ is the discriminant of f and $\square$ a polynomial function of the variables $\lambda_2, \lambda_3, \pi_4, \pi_5$. Observe that Σ_2 is almost a *universal* separating element, i.e. if f is separable, then its Lagrange resolvent $\mathcal{L}_{\Theta_{(0,1,0,0,0,0)}}$ is still separable, provided that π_5 is non zero, which is insured by the extra hypothesis of irreducibility of f.

This resolvent is known as the *Cayley* resolvent, and this result was proved by Arnaudiès–Valibouze for the general irreducible polynomial of degree 5, but only over the rational numbers.

Stability and Bifurcation Analysis of Coupled Fitzhugh-Nagumo Oscillators

William Hanan[1], Dhagash Mehta[1], Guillaume Moroz[2*], Sepanda Pouryahya[1]

[1] National University of Ireland, Maynooth,
Mathematical Physics Department, National University of Ireland Maynooth, Maynooth, Co. Kildare, Ireland
wgh@thphys.nuim.ie, dmehta@thphys.nuim.ie, sepour@thphys.nuim.ie

[2] Institut de Recherche en Communications et Cybernétique de Nantes UMR CNRS 6597, 1, rue de la Noe, BP 92101, 44321 Nantes Cedex 03 France
Guillaume.Moroz@irccyn.ec-nantes.fr

.....

Abstract

Neurons are the central biological objects in understanding how the brain works. The famous Hodgkin-Huxley model, which describes how action potentials of a neuron are initiated and propagated, consists of four coupled nonlinear differential equations. Because these equations are difficult to deal with, there also exist several simplified models, of which many exhibit polynomial-like non-linearity. Examples of such models are the Fitzhugh-Nagumo (FHN) model, the Hindmarsh-Rose (HM) model, the Morris-Lecar (MR) model and the Izhikevich model. In this work, we first prescribe the biologically relevant parameter ranges for the FHN model and subsequently study the dynamical behaviour of coupled neurons on small networks of two or three nodes. To do this, we use a computational real algebraic geometry method called the Discriminant Variety (DV) method to perform the stability and bifurcation analysis of these small networks. A time series analysis of the FHN model can be found elsewhere in related work [11].

The Fitzhugh-Nagumo Model The Fitzhugh-Nagumo (FHN) system of equations is a prototypic model of excitability. Introduced by FitzHugh [9] with equivalent circuit by Nagumo et al.[19] the system is a generalization of the Van der Pol oscillator [7] and a reduction of the neural electrophysiological model due to Hodgkin and Huxley (HH) [12]. As a generic model of excitability and oscillatory dynamical behavior, the FHN system is of relevance for a range of physical and physiological research topics[1, 8, 15, 17, 18, 21, 24], the most extensive of which are those concerned with cardiac [2, 10, 16, 22, 25] and neuronal [4, 23, 28, 31] cell dynamics. The corresponding equations for coupled FHN neurons are

$$\begin{aligned} \frac{dx_i}{dt} &= x_i - \frac{x_i^3}{3} - y_i + g\sum_{j=1}^{n}(x_i - x_j) \\ \frac{dy_i}{dt} &= \epsilon(x_i + a - b\,y_i), \end{aligned} \tag{1}$$

for $i = 1, \ldots, n$ and where we have taken $a \in [-2, 2]$, $b \in (0, \infty)$, $-1 \leq g \leq 1$ ($g \neq 0$) and $0 < \epsilon \leq 0.1$. It is well-known that in the FHN model, the variables have no direct physiological interpretation. However, for the parameter ranges quoted above, the qualitative behaviour

†Correspondence to: IRCCyN, 1, rue de la Noe, BP 92101, 44321 Nantes Cedex 03 France. Tel : +33 2 40 37 69 00. Fax : +33 2 40 37 69 30

of the x's and y's are similar to that of the voltage and gating variables in the Hodgkin-Huxley equations.

Stationary points in the FHN system Neurons within the central nervous system can exist in a variety of dynamical states. Many, for example, are in a state of quiescence and elicit a relaxation oscillation, known as an *action potential*, when perturbed with a suprathreshold stimulus. This is oft termed as "spiking". With the appropriate choice of parameters, the FHN model can exhibit such behaviour.

However, upon varying these parameters, one can also invoke a *Hopf bifurcation* resulting in relaxation oscillations at an intrinsic frequency without the need for any stimulation. Such self sustained neurons can be found within the central nervous system and can have complex interactions with the environment. A typical example is that of *circadian cells* which act like the organisms clock cells and have been shown to have the ability to entrain their oscillations with the environments light-dark cycle [29].

Neurons have also been shown to express bistability [26, 27], which again is present in the FHN model due to the cubic nonlinearity present in the system.

The dynamical states in the FHN model described above can be identified via a stability analysis of the stationary points in the system corresponding to $\frac{dx_i}{dt} = \frac{dy_i}{dt} = 0$. Such an analysis is thus essential if one is to have a basic understanding of the system. In fact, the steady state equations of this model have already been solved exactly for the $n = 1$ and $n = 2$ cases [5].

Below we use the DV method to solve the corresponding equations for the $n = 3$ case and study the stability and bifurcation structure of these systems. The functions we use are in the packages `Groebner` and `RootFinding[Parametric]` of the computer algebra system `Maple 13`. Due to the polynomial-like nonlinearity also exhibited in the HM, MR and Izhikevich models, the same technique may also be applied to these systems.

The problem adresssed in this paper can also be related to the more general problem of the algebraic analyisis of the solutions of a differential system, studied for example in [3], [20],[30] or [32],[33].

Also, even if we didn't use them, other software address some of the computations (quantifier elimination, sample points extraction) that we are doing with `Maple`. In particular, it is worth mentionning `Discoverer`*, QEPCAD†, `REDUCE`‡, `Mathematica`§, or `RAG`¶.

Agebraic tools For this study, we used the Discriminant Variety ([14]) and the Cylindrical Algebraic Decomposition ([6]). The Discriminant Variety is an implicit representation of the desired partition, and the Cylindrical Algebraic Decomposition describes explicitly each cell of the partition.

Combined together, these 2 methods are well adapted the analysis of the steady and stable points. They provide a partition of the parameter space in connected cells, such that within each cell, the number of steady state (resp. stable state) is constant.

Finally, the main algebraic criterion to decide if a solution is stable is the Routh-Hurwitz criterion, or its Liénard-Chipart variant ([13]). For our problem, we used a reduced criterion, more adapted to the algebraic DV and CAD methods.

*http://www.is.pku.edu.cn/ xbc/discoverer.html, developped by Wang, Xia et al.
†http://www.usna.edu/Users/cs/qepcad/B/QEPCAD.html, developped by Hong et al.
‡http://reduce-algebra.sourceforge.net/, developped by Hearn, Weispfenning et al.
§http://www.wolfram.com/products/mathematica/index.html
¶http://www-spiral.lip6.fr/ safey/RAGLib/distrib.html, developped by Safey El Din

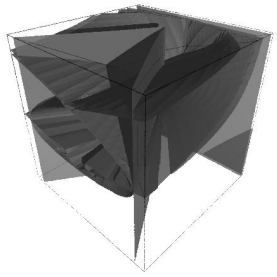


Figure 1: Parameter space decomposition adapted to the number of steady states for $n = 3$: in each connected component outside the surface, there is a constant number of steady state.

g	$]-\infty,\mu_1[$	$]\mu_1,\mu_2[$	$]\mu_2,\mu_3[$	$]\mu_3,\mu_4[$	$]\mu_4,\mu_5[$	$]\mu_5,\mu_6[$
Steady states	3	3	3	3	3	3
Stable states	2	2	2	2	2	2

g	$]\mu_6,\mu_7[$	$]\mu_7,\mu_8[$	$]\mu_8,\mu_9[$	$]\mu_9,\mu_{10}[$	$]\mu_{10},\mu_{11}[$	$]\mu_{11},\mu_{12}[$
Steady states	3	3	3	3	3	15
Stable states	2	2	2	2	2	2

g	$]\mu_{12},\mu_{13}[$	$]\mu_{13},\mu_{14}[$	$]\mu_{14},\mu_{15}[$	$]\mu_{15},\mu_{16}[$	$]\mu_{16},\mu_{17}[$	$]\mu_{17},\mu_{18}[$
Steady states	15	15	15	15	27	27
Stable states	2	2	2	2	8	8

g	$]\mu_{18},\mu_{19}[$	$]\mu_{19},\mu_{20}[$	$]\mu_{20},\mu_{21}[$	$]\mu_{21},\mu_{22}[$	$]\mu_{22},\mu_{23}[$	$]\mu_{23},\mu_{24}[$
Steady states	27	27	27	27	27	27
Stable states	8	8	8	14	6	6

g	$]\mu_{24},\mu_{25}[$	$]\mu_{25},\mu_{26}[$	$]\mu_{26},\mu_{27}[$	$]\mu_{27},\infty[$
Steady states	27	27	15	15
Stable states	6	6	6	6

where:
$\mu_1 \approx -1.690122506$, $\mu_2 \approx -.3010608116$, $\mu_3 \approx -.2787081900$, $\mu_4 \approx -.2530108085$,
$\mu_5 \approx -.2200916671$, $\mu_6 \approx -.2119432516$, $\mu_7 \approx -.1972651791$, $\mu_8 \approx -.1944444444$,
$\mu_9 \approx -.1797585308$, $\mu_{10} \approx -.1768496368$, $\mu_{11} \approx -.1666666667$, $\mu_{12} \approx -.1111111111$,
$\mu_{13} \approx -.09533095509$, $\mu_{14} \approx -.08765527212$, $\mu_{15} \approx -.08093319477$, $\mu_{16} \approx -.06753407911$,
$\mu_{17} \approx -.06313592910$, $\mu_{18} \approx .1646239760$, $\mu_{19} \approx .2210124208$, $\mu_{20} \approx .2847525372$,
$\mu_{21} \approx .3136620682$, $\mu_{22} \approx .3333333333$, $\mu_{23} \approx .3372096920$, $\mu_{24} \approx .3373316121$,
$\mu_{25} \approx .3483417636$, $\mu_{26} \approx .3562092137$, $\mu_{27} \approx .3888888889$.

Figure 2: Number of steady and stable states for $a = 0$, $b = 2$ according to the parameter g

Results This model was studied and for the case $n = 2$ in [5]. We focused our work on the case $n \geq 3$.

First, we succeeded in describing completely the steady states. The result of our computation is summarized in Figure 1.

However, when $n \geq 3$, the computations of the stable states are much more difficult and we did not succeed in describing the full parameter space according to the number of stable states. Nevertheless, by fixing the 2 parameters a and b to the numerical values (respectively 0 and 2), we managed to described the number of steady and stable states according to the free parameter g, in the case $n = 3$. Since there is only 1 parameter, the description of the parameter space is an union of interval in the variable g for which the number of stable (and steady) solutions is constant.

The results of the computations are summarized in Figure 2.

Acknowlegement DM was supported by Science Foundation of Ireland.
SP was supported by the Irish Research Council for Science Engineering and Technology (IRCSET).

References and Notes

[1] S. Barland, O. Piro, M. Gludici, J.R. Tresicce, and S. Balle. Experimental evidence of van der pol-fitzhugh-nagumo dynamics in semiconductor optical amplifiers. *Phys. Rev. E*, 68(3):036209, 2003.

[2] V. N. Biktashev, A. V. Holden, S. F. Mironov, A. M. Pertsov, and A. V. Zaitsev. Three-dimensional organisation of re-entrant propagation during experimental ventricular fibrillation. *Chaos Sol. & Frac.*, 13(8):1713–1733, 2002.

[3] F. Boulier, M. Lefranc, F. Lemaire, P.-E. Morant, and A. Ürgüplü. On proving the absence of oscillations in models of genetic circuits. In *AB*, pages 66–80, 2007.

[4] David Brown, Jianfeng Geng, and Stuart Feerick. Variability of firing of hodgkin-huxley and fitzhugh-nagumo neurons with stochastic synaptic input. *Phys. Rev. Lett.*, 82(23):4731–4734, 1999.

[5] Sue Ann Campbell and Michael Waite. Multistability in coupled fitzhugh–nagumo oscillators. *Nonlin. Anal.*, 47:1093–1104, 2001.

[6] George E. Collins. *Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition*. Springer Verlag, 1975.

[7] B. Van der Pol. A theory of the amplitude of free and forced triode vibrations. *Radio Review*, 1:701–710, 754–762, 1920.

[8] Orazio Descalzi, Jaime Cisternas, Daniel Escaff, and Helmut R. Brand. Noise induces partial annihilation of colliding dissipitive solitons. *Phys. Rev. Lett.*, 102(18):188304, 2009.

[9] R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical J.*, 1(6):445–466, 1961.

[10] R. A. Gray. Termination of spiral wave breakup in a fitzhugh-nagumo model via short and long duration stimuli. *Chaos*, 12(3):941–951, 2002.

[11] William Hanan, Dhagash Mehta, Sepanda Pouryahya, and Julien Sprott. Work in progress.

[12] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (Lond.)*, 117:500–544, 1952.

[13] P. Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, New York, 1985.

[14] Daniel Lazard and Fabrice Rouillier. Solving parametric polynomial systems. *J. Symb. Comput.*, 42(6):636–667, 2007.

[15] Anita Shu-Han Lin, Martin L. Buist, Nicolas P. Smith, and Andrew J. Pullan. Modelling slow wave activity in the small intestine. *J. Theo. Bio.*, 242(2):356–362, 2006.

[16] A.F.M Marée and A.V. Panfilov. Spiral breakup in excitable tissue due to lateral instability. *Phys. Rev. Lett.*, 78(9):1819–1822, 1997.

[17] F. Marino, M. De Rosa, and F. Marin. Canard orbits in fabry-perot cavities induced by radiation pressure and photothermal effects. *Phys. Rev. E*, 73:026217, 2006.

[18] Bradley Marts, David J. W. Simpson, Aric Hagberg, and Anna L. Lin. Period doubling in a periodically forced belousov-zhabotinsky reaction. *Phys. Rev. E*, 76:026213, 2007.

[19] J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proc. IRE*, 50(10):2061–2070, 1962.

[20] Wei Niu and Dongming Wang. Algebraic approaches to stability analysis of biological systems. *Mathematics in Computer Science*, 1(3):507–539, 2008.

[21] A. Pereira, P. M. J. Trevelyan, U. Thiele, and S. Kalliadasis. Interfacial instabilities driven by chemical reactions. *Euro. Phys. J. Spec. Top.*, 166:121–125, 2009.

[22] A.M. Pertsov, J.M. Davidenko, R. Salomonsz, W.T. Baxter, and J. Jalife. Spiral waves of excitation underlie reentrant activity in isolated cardiac-muscle. *Circ. Res.*, 72(3):631–650, 1993.

[23] W. Rall, R.E. Burke, W.R. Holmes, J.J.B. Jack, S.J. Redman, and I. Segev. Matching dendritic neuron models to experimental-data. *Physiological Reviews*, 72(4):S129–S186, 1992.

[24] Hiroto Shoji, Kohtaro Yamada, and Takao Ohta. Interconnected turing patterns in three dimensions. *Phys. Rev. E*, 72:065202(R), 2005.

[25] S. Sinha, K. M. Stein, and D. J. Christini. Critical role of inhomogeneities in pacing termination of cardiac reentry. *Chaos*, 12(3):893–902, 2002.

[26] I. Tasaki. Demonstration of two stable states of the nerve membrane in potassium-rich media. *J. Physiol. (Lond.)*, 148(2):306–331, 1959.

[27] Ichiji Tasaki and Susumu Hagiwara. Demonstration of two stable potential states in the squid giant axon under tetraethylammonium chloride. *J. Gen. Physiol.*, 40:859–885, 1957.

[28] Rail Toral, C. Masoller, Claudio R. Mirasso, M. Ciszak, and O. Calvo. Characterization of the anticipated synchronization regime in the coupled fitzhugh-nagumo model for neurons. *Physica A*, 325(1-2):192–198, 2003.

[29] Shobi Veleri, Christian Brandes, Charlotte Helfrich-F́orster, Jeffrey C. Hall, and Ralf Stanewsky. A self-sustaining, light-entrainable circadian oscillator in the drosophila brain. *Curr. Biol.*, 13(20):1758–1767, 2003.

[30] Dongming Wang and Bican Xia. Stability analysis of biological systems with real solution classification. In *ISSAC '05: Proceedings of the 2005 international symposium on Symbolic and algebraic computation*, pages 354–361, New York, NY, USA, 2005. ACM.

[31] D.Q. Wei, X.S. Luo, and Y.L. Zou. Firing activity of complex space-clampled fitzhugh-nagumo neural netowrks. *Eur. Phys. J. B*, 63:279–282, 2008.

[32] H. Yoshida, K. Nakagawa, H. Anai, and K. Horimoto. An algebraic-numeric algorithm for the model selection in kinetic networks. In *CASC*, pages 433–447, 2007.

[33] H. Yoshida, K. Nakagawa, H. Anai, and K. Horimoto. Exact parameter determination for parkinson's disease diagnosis with pet using an algebraic approach. In *AB*, pages 110–124, 2007.

Computer Algebra for Integer Portfolio problems

F.J. Castro-Jiménez[1], M.J. Gago-Vargas[2], M.I. Hartillo[3], J. Puerto[4] and J.M. Ucha[5*]

[1 2 5] Dpto. Álgebra, Universidad de Sevilla
castro@us.es, gago@algebra.us.es, ucha@us.es

[3] Dpto. Matemática Aplicada, Universidad de Sevilla
hartillo@us.es

[5] Dpto. Estadística e Inv. Operativa, Universidad de Sevilla
puerto@us.es

Abstract

Optimally management of portfolio risk is an essential component of modern asset allocation. The classical mean-variance approach of Markowitz is enhanced if integer variables can be involved.

We propose an algebraic approach to maximize the expected return of a portfolio under a given admissible level of risk measured, in principle, by the variance. To obtain an exact solution, it is an essential ingredient the computation via Gröbner basis of test-sets corresponding to linear subproblems. These test-sets are used in a strategy similar to the one introduced in [9].

1 Motivation

Mean-variance portfolio construction lies at the heart of modern asset management and has been among the most investigated fields in the economic and financial literature. The classical Markowitz's approach (cf. [7]) rests on the presumption that investors choose among n risky assets and look for the corresponding weights $w_1, \ldots, w_n$ of each asset in their portfolios, on the basis of 1) previously estimated expected returns $\mu_1, \ldots, \mu_n$ of each asset, and 2) on the corresponding risk of the portfolio measured by the covariance matrix Ω.

Portfolios are considered *mean-variance efficient* if they maximize the expected return for a given admissible risk (variance) r^2:

$$\begin{aligned} \max \quad & \mu_1 w_1 + \cdots + \mu_n w_n, \\ \text{subject to} \quad & (w_1, \ldots, w_n)\Omega \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} \leq r^2, \\ & w_1 + \cdots + w_n = 1, \\ & w_i \in \mathbb{R}, \end{aligned}$$

or, equivalently, if they minimize the variance for a given admissible return R (this is the usual presentation of the problem in the literature).

*Correspondence to: Facultad de Matemáticas, Avda. Reina Mercedes s/n. Sevilla 41012 (Spain). Fax +34 954556938

It is more realistic to consider integer variables to present the problem taking into account the prices $a_1, \ldots, a_n$ of the products, variables $x_1, \ldots, x_n$ for the quantities of each product and the total available budget B of the investor (as in [4]):

$$\begin{aligned} \max \quad & \textstyle\sum_{i=1}^{r} \mu_i x_i \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{r} a_i x_i \leq B, \\ & Q(\boldsymbol{x}) = \tfrac{1}{B^2}(a_1 x_1, \ldots, a_n x_n)\Omega \begin{pmatrix} a_1 x_1 \\ \vdots \\ a_n x_n \end{pmatrix} \leq r^2, \\ & x_i \in \mathbb{Z}^+. \end{aligned} \tag{1}$$

Although the standard statement of the mean-variance portfolio problem uses continuous variables, there are different reasons to consider integer variables (apart from the problems of rounding continuous solutions) as the existence of transaction costs, bounds on the number of assets, or the constraint of buying stocks by lots (cf. [10], [6] or [4]).

2 Previous works and our approach

The integer portfolio problem has been treated with different methods in [2], [8] or [10], and more recently in [4] and [6]. The approach of [1] could be applied in principle, but it is a work in progress to do it in an effective way.

Our approach to obtain an exact solution for Problem (1) rests in the following fact: we have a linear objective function and constraints that are linear and non-linear, as in the problem treated in [9]. Generalizing the ideas of [9], to solve an integer programming problem $\mathbf{P}$ with linear objective function under linear and non-linear constraints, a general method can be applied:

1. First obtain a *test-set* for the linear part, let us call it $\mathbf{P'}$, of $\mathbf{P}$. A *test-set* T of $\mathbf{P'}$ is a set of vectors in $\mathbb{Z}^n$ which verifies the following condition: given a *feasible point* $\boldsymbol{p}$ of $\mathbf{P'}$ (that is, a point for which the linear conditions hold), if none of the feasible points obtained adding the elements of T to $\boldsymbol{p}$ improve the value of the objective function, then $\boldsymbol{p}$ is an optimum of $\mathbf{P'}$. A test-set of a linear integer problem can be obtained via Gröbner bases (the seminal work is [3]). A good way to obtain these bases is using `4ti2` [5].

2. Use the corresponding *reversal* test-set —the one whose vectors decrease the objective function— to travel into the set (tree) of feasible points of $\mathbf{P'}$ until you obtain feasible points *for the whole problem* $\mathbf{P}$ (in our case, it means, portfolios with admissible risk). If this happens, one can prune the remaining feasible solutions in this branch of the tree of solutions of $\mathbf{P'}$.

Our approach consists of applying this general idea to Problem (1) or, more precisely, to the original problem with some extra linear restrictions. This is necessary in order to avoid the huge regions you need to explore with the reversal test-set for real examples. The required initial data is:

- The best expected return R_c of the continuous relaxation of the problem.

- The R_e return associated with a discrete feasible point $\boldsymbol{p}_e$.
- Solving the continuous problems

$$\begin{array}{rc} & \min x_i \\ \text{s.t.} & \sum_{j=1}^n a_j x_j \leq B, \\ & R_e \leq \sum_{j=1}^n \mu_j x_j \leq \lfloor R_c \rfloor, \\ & \frac{1}{B^2} Q(\boldsymbol{x}) \leq r^2, \qquad x_j \in \mathbb{R}, j = 1, \ldots, n, \end{array}$$

  we obtain lower bounds b_i for each variable x_i, applying the ceiling function. Upper bounds of the continuous problem can be computed but their introduction produces much bigger Gröbner bases. In the examples, we have used them to delete components and reduce the dimensionality of the problem when lower and upper bounds match.
- Add, iteratively, some extra hyperplanes (quasi)-tangent to the region defined by the admissible risk (so they do not eliminate points with admissible risk).

Then, usually Problem (1) produce a problem of the folowing type:

$$\begin{array}{rc} & \max \sum_{i=1}^n \mu_i x_i \\ \text{s.t.} & \sum_{i=1}^r a_i x_i \leq B, \\ & R_e \leq \sum_{i=1}^n \mu_i x_i \leq \lfloor R_c \rfloor, \\ & \boldsymbol{n}_k^t \boldsymbol{x} \leq c_k, k = 1, \ldots, s, \\ & x_i \geq b_i, i = 1, \ldots, n \\ & \frac{1}{B^2} Q(\boldsymbol{x}) \leq r^2, \end{array}$$

where $\boldsymbol{n}_k^t \boldsymbol{x} = c_k, k = 1, \ldots, s$ are the new (quasi)-tangent hyperplanes. As soon as a new feasible point $\boldsymbol{p}'_e$ is found that improves the value of R_e, the problem can be reformulated and we are closer to the optimum

Remark 2.1 *If a new feasible point is not produced by the search in a reasonable time, branch-and-cut techniques have been used successfully.*

3 Examples

We have been able to treat real examples of $n = 35, 40, 50$ stocks in a reasonable CPU running time: it takes less than 2 minutes to compute the Gröbner bases of the linear part and less than 5 minutes to finish the total process.

For a particular example* we have obtained the following table. The column r_0^2 is the admissible risk, $r_{\max}^2$ is the maximum risk over the polytope defined by the linear restrictions of Problem (1), the number of tangents added, the number of elements in the Gröbner bases and the number of points processed.

*In this case, data form 44 stocks from Eurostoxx index, from January 2003 to December 2007. The vector of initial prices is given by the prices of the stocks on January 3rd 2008.

r_0^2	$r_{\max}^2$	tangents	basis	processed	optimum
0.0015	0.00154	1	6657	165	$x_6 = 42, x_{14} = 4, x_{16} = 29,$ $x_{20} = 5, x_{28} = 1, x_{36} = 8.$
0.0020	0.00205	1	40256	137	$x_6 = 51, x_{14} = 5, x_{16} = 19,$ $x_{20} = 1, x_{28} = 1, x_{36} = 12.$
0.0025	0.00256	2	20782	32	$x_6 = 59, x_{14} = 6, x_{16} = 13,$ $x_{28} = 2, x_{36} = 10.$
0.0030	0.00301	1	12504	62	$x_6 = 64, x_8 = 1, x_{14} = 8, x_{16} = 1,$ $x_{28} = 1, x_{36} = 10, x_{37} = 1$
0.0035	0.00351	1	2357	0	$x_6 = 68, x_{14} = 10, x_{16} = 1, x_{36} = 5.$
0.0040	0.00430	0	569	9904	$x_6 = 74, x_{14} = 10, x_{16} = 1,$ $x_{28} = 2, x_{36} = 1.$
	0.00404	1	11924	11	$x_6 = 74, x_{14} = 10, x_{16} = 1,$ $x_{28} = 2, x_{36} = 1.$
0.0045	0.00451	1	7087	0	$x_6 = 84, x_{14} = 6, x_{16} = 1,$ $x_{28} = 1.$
0.0050	0.00508	0	357	6	$x_6 = 91, x_{14} = 3, x_{28} = 1.$

References and Notes

[1] Bertsimas, D., G. Perakis, G. and Tayur, S. A new algebraic geometry algorithm for integer programming. Management Science, 46, 999-1008, 2000.

[2] Bienstock, D. Computational study of a family of mixed-integer quadratic programming problems. Mathematical Programming 74, 121–140, 1996.

[3] Conti, P. and Traverso, C. Buchberger Algorithm and Integer Programming. AAECC 1991: 130-139.

[4] Corazza, Mario and Favaretto, Daniela. On the existence of solutions to the quadratic mixed-integer mean-variance portfolio selection problem. European Journal of Operational Research 176, 1947–1960, 2007.

[5] Hemmecke, R., Hemmecke, R. and Malkin, P. N. 4ti2 version 1.2. Computation of Hilbert bases, Graver bases, toric Gröbner bases, and more, Available at `http://www.4ti2.de/`, 2005.

[6] Li, Han-Lin and Tsai, Jung-Fa. A distributed computation algorithm for solving portfolio problems with integer variables. European Journal of Operational Research 186, 882–891, 2008.

[7] Markowitz, H. Portfolio selection. Journal of Finance 7, 77–91. 1952.

[8] Sun, X.L. and Li, D. On the relationship between the integer and continuous solutions of convex programs. Operations Research Letters 29, 87–92, 2001.

[9] Tayur, R.Tayur, Thomas, Rekha R. and Natraj, N.R. An algebraic geometry algorithm for scheduling in presence of setups and correlated demands. Mathematical Programming 69, 369–401, 1995.

[10] Young M.R., A minimax portfolio selection rule with linear programming solution. Management Science 44(5), 673–683, 1998.

Algebraic points in geometry and application to CAD

DANIEL LAZARD*

Affiliation 1: UPMC, Univ Paris 06, LIP6, France
Affiliation 2: INRIA Paris-Rocquencourt, SALSA project team, France
Affiliation 3: CNRS, LIP6, France

Abstract

A point is is usually represented, in computational geometry, by the vector of its coordinates. But when computing with curved objects, these coordinates become real algebraic numbers, which makes computation very difficult.

The efficiency of existing solvers for real algebraic systems allows to replace this usual representation by a system of equations and inequalities which has exactly one solution, completed by an interval approximation of the coordinates. This allows to implement every basic operation on points using at most one call to a real solver. The operations, like sorting, which are efficient without degenracies with floating point approximations remain possible with usual interval arithmetic.

To illustrate the possibilities of this representation of points, it is shown how it may be used for Cylindrical Algebraic Decomposition (CAD). A comparison with the usual implementation of CAD is done on a classical difficult example of quantifier elimination.

1 Introduction

Usually, in computational geometry, *points* are represented by the vector of their coordinates. This is fine when these coordinates are floating point numbers, which means that the points are approximately defined. This is also fine when working only with linear objects (points, lines and planes): in this case, if the input consists in points with rational coordinates, all the points which are computed have also rational coordinates and the computation may proceed exactly without problem of implementation.

On the other hand modern computational geometry has to compute with curved objects and requires frequently to use exact computation when encountering degenerate situations, where floating point computation may lead to wrong results. Unfortunately, the points which are constructed with curved objects are generally not rational, but algebraic. The simplest example is the intersection of the circle of equation $x^2 + y^2 = 1$ with the line $x = y$, which consists in two points of coordinates $\pm\sqrt{2}$. Thus exact computations with the usual model of points represented by their coordinates need an arithmetic of real algebraic numbers. Such an arithmetic is very difficult to implement (I do not know any which is available) and extremely slow. Moreover, efficiency implies to use floating point computation or interval arithmetic as long as possible and to use exact arithmetic only when needed.

It follows from above remarks that another model is needed to represent points in geometry. Our proposal consists in representing a point by a record consisting in a system of equations and inequalities which have a unique real solution and in completing this by approximate values (floating point or interval) for the coordinates. Efficient software exist to solve polynomial systems (function ROOTFINDING[ISOLATE] in Maple). They allow to do easily every usual operations on the points and to provide exact answers. On other hand for most operations, like sorting, the usual floating point representation remains available.

*Correspondence to: LIP6, UPMC, F-75252, Paris, France and mailto: Daniel.Lazard@lip6.fr

This representation of points is not really new, as it appears (and is used), more or less implicitly, in several papers (citations will be added in a final version). However, it is a tool which could be used for many questions of computational geometry. It seems therefore useful to publish it for itself, which is the object of the present paper.

In the following sections we describe first the specification which are needed for a solver and precise our representation of the points; this includes a description of the implementation of the main basic operations on points (Section 2). Then we describe how this may be used to compute a CAD (Cylindrical Algebraic Decomposition) (Section 3). Finally we present how it applies to a classical problem of quantifier elimination (Section 4).

2 Geometrical points

As said above, we represent a point in $\mathbb{R}^n$ by a record of three fields called EQS, INEQ and APPROX. These fields have the following specification.

• EQS is a set of polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ which have a finite number of common zeros in $\mathbb{C}^n$.

• EQS is a set of polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ which have a INEQ is a set of polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ such that the system $\{f = 0$ for f in EQS, $g > 0$ for g in INEQ$\}$ has exactly one real solution which are the coordinates of the point.

• APPROX is a list of interval approximationsof the coordinates of the point whose elements have, for example, the shape $x_i = [a_i, b_i]$. For technical reasons, it is also required that a_i and b_i are either both positive or both negative or both null. In other words, if an interval contains 0 then it is $[0, 0]$.

The fields of the representation of a point p will be denoted by EQS(p), INEQ(p) and APPROX(p).

For a point with rational coordinates this representation is not essentially different from the usual one. In fact, for a point p with rational coordinates $a_1, \ldots a_n$, we have EQS$(p) = \{x_1 - a_1, \ldots, x_n - a_n\}$, INEQ$(p) = \{\}$ and APPROX$(p) = [x_1 = [a_1, a_1], \ldots, x_n = [a_n, a_n]]$.

Some functions are needed to allow working with this definition of a point, the first one being a tool to verify that the specification of a point is satisfied, to compute the field APPROX and, if needed, to refine this approximation. All these auxiliary functions are based on the function RSOLVE for solving real systems of equations and inequalities, which we describe now.

The function RSOLVE(EQUATIONS, INEQUALITIES) has the following specification.

• EQUATIONS is a set of polynomials with rational coefficients which are interpreted as equations $f = 0$ for f in EQUATIONS.

• INEQUALITIES is a set of polynomials with rational coefficients which are interpreted as inequalities $g > 0$ for g in INEQUALITIES. Any variable appearing in INEQUALITIES should appear also in EQUATIONS.

• If the system of equations defined by EQUATIONS has infinitely many complex solutions the output in the text "non zero-dimensional system".

• In the other case, the output is a list of the real solutions of the system of equations and inequalities defined by EQUATIONS and INEQUALITIES, each solution being represented by a set of equations of the form $x = [a, b]$, where x is the name of a variable appearing in EQUATIONS and $[a, b]$ is an interval containing the value of x at the solution. Thus each each solution is represented by a box containing it in $\mathbb{R}^n$. There are two additional requirements in this output specification: The boxes of two different solutions do not intersect and the end points of every interval have the same sign. In other words, the boxes separate the solutions and any interval which contains 0 is equal to $[0, 0]$.

An optional parameter, called PREC, may be needed for the function RSOLVE (EQUATIONS, INEQUALITIES): It is a upper bound for the length of the intervals in the output.

As far as we know, RSOLVE is not directly implemented in existing software. However, it is easy to construct it from Maple function ROOTFINDING[ISOLATE] whose specification is exactly the same as that of RSOLVE when INEQUALITIES is empty. When a bug of releases 10 to 12 of MAPLE will be corrected, it will suffices, to implement RSOLVE, to select the desired solutions from the output of ISOLATE(EQUATIONS, CONSTRAINTS=INEQUALITIES, OUTPUT="INTERVAL") or ISOLATE(EQUATIONS, CONSTRAINTS=INEQUALITIES, OUTPUT="INTERVAL", DIGITS=D). To avoid this bug, the easiest way consists presently in constructing the list INEQ2EQ:= $[X[g] - g$ FOR g IN INEQUALITIES] where $X[g]$ is a new variable (one for each g). Then, if EQS2 is the concatenation of EQUATIONS and INEQ2EQ the call of ISOLATE(EQS2, OUTPUT="INTERVAL") allows to produce the desired output: it suffices to remove the solutions for which some $X[g]$ is not positive and the values of $X[g]$ in the remaining solutions. The drawback of this patch is to introduce a number of new variables which makes the computation slower than it should be. However, in many cases, the interval evaluation of the elements INEQUALITIES on the output of ISOLATE(EQUATIONS, OUTPUT="INTERVAL") produces non intersecting intervals and this allows to avoid the introduction of the variables $X[g]$.

Function RSOLVE, allows to implement the basic operations on points.

Testing correctness of a point and computing field APPROX **to a desired precision.** If p is the point to test, it suffices to call RSOLVE(EQS(p), INEQ(p), PREC = SOMEVALUE): The point is correct if the output is reduced to a single element and, in this case, the output gives the field APPROX.

Equality of two points. The points p and q are equal if and only if the output of RSOLVE (EQS(p) $\cup$ EQS(q), INEQ(p) $\cup$ INEQ(Q)) has a non empty output.

Inclusion of a point in an algebraic variety. Let p be a point and VAR be a set of implicit equations of a variety. The point belongs to the variety if and only if RSOLVE(EQS(p) $\cup$ VAR, INEQ(p)) has a non empty output.

Sign of a polynomial at a point. This operation is frequently needed, for example for convex hull computation or for testing if a point is inside a conic. In most cases an interval evaluation suffices (using the field APPROX). However, if the resulting interval contains zero, an exact computation may be needed which is described now. Let p be the point and POL be the polynom. We introduce a new variable X and run RSOLVE(EQS(p) $\cup$ $X-$POL, INEQ(p)). The specification of RSOLVE asserts that the sign $+$, $-$ or 0 of POL at the unique solution may be immediately deduced from the interval of the values of X.

Comparing the values of a polynomial at two points. This operation is needed to sort a set of points with respect to one of their coordinates. Let p and q be the points and POL be the polynomial. As above, if the evaluation of POL at APPROX(p) and APPROX(q) provides non intersecting intervals, the comparison is immediate. If it is not the case one may proceed as follows. Let q' be the data structure obtained by substituting $x_1, \ldots, x_n$ by $y_1, ..., y_n$ in q and POL$'$ be the result of the same substitution in POL; let X be an auxiliary new variable. Then the sign of X in the output of RSOLVE(EQS(p) $\cup$ EQS(q') $\cup$ $X-$(POL$-$POL$'$), INEQ(p) $\cup$ INEQ(q')) gives the result. In fact, this amounts to compute the sign of POL$-$POL$'$ at a point in $\mathbb{R}^{2n}$ defined by the pair (p, q).*

The drawback of this operation is to apply RSOLVE with twice the number of variables than the other operations on the points. This may be time consuming. However, in some specific context, one may avoid this doubling of the number of variables.

This is the case when one knows that the equality of the values of POL implies the equality of the points, for example if the points to be compared belong to a fixed line. This is especially the

*It should be noted that this will be applied in next section to points which have not the same number of coordinates.

case in the lifting step of CAD (see Section 3), where the last coordinate has to be compared for points having the same $n-1$ first coordinates.

In this situation, one may proceed as follows. If comparison results of the interval evaluation of POL, its OK. If not test the equality of the points. It there are equal, this is OK. Else increase the precision of the approximation until interval evaluation gives the result.

Converting the output of RSOLVE **to a list of points.** The main way to construct points in geometry is as the intersection of two curves in the plane or as the intersection of a surface and a curve in the space. Thus almost all points are constructed from the output of RSOLVE.

More precisely, let us consider the output of a call to RSOLVE(EQUATIONS, INEQUALITIES), which consists in a finite list $[\text{SOL}_1, \ldots, \text{SOL}_k]$ of solutions. To convert these solutions into points p_i, we take $\text{APPROX}(p_i) = \text{SOL}_i$, $\text{EQS}(p_i) = \text{EQUATIONS}$ and $\text{INEQ}(p_i) = \text{INEQUALITIES} \cup \text{MORE}_i$ where MORE_i is a set of additional constraints, we define now.

If there is only one solution ($k = 1$) then MORE_1 is empty. In the other case, MORE_i is intended to separate the i-th solution from the others. One could take the condition that each coordinate belongs to the interval defining it in the solution. This is not a good idea: in the subsequent call of RSOLVE, it will have to decide of the sign of quantities which are close to zero; this will make the computation very slow and produce very small intervals in the output, whose bounds are rational number with huge numerators and denominators or floating point numbers with very high precision.

This problem may be solved in the following way. Suppose that there is is a variable x_i for which the intervals $x_i = [a_{i,j}, b_{i,j}]$ for $j = 1, \ldots, k$ are pairwise disjoint. By sorting and reindexing the solutions one may suppose that $b_{i,j} < a_{i,j}$. Choosing a rational number r_j with numerator and denominator as small as possible in the interval $]b_{i,j}, a_{i,j+1}[$ for $i = 2, \ldots, k-1$, it suffices to choose $\text{MORE}_j = \{r_j - x_i, x_i - r_{j-1}\}$ for $j = 2, \ldots, k$, $\text{MORE}_1 = \{r_1 - x_i\}$ and $\text{MORE}_k = \{x_i - r_{k-1}\}$. If there is no variable for which the intervals are pairwise disjoint, one may use, instead of a variable, a random linear combination of the variables for which the intervals are pairwise disjoint; this is the case (with probability one) after increasing, if needed, the precision PREC. One may also use the first variable to separate, as much as possible the solutions, then use the second variable to separate more solutions, and so on. The specification of RSOLVE implies that, eventually, this process separates all the solutions.

Projection of a point on a subspace. Gröbner basis algorithm allows to compute the equations of a projection. However these equations are of high degree (usually this degree is the $i+1$-th power of the degree of the equations before projection, if the projection eliminates i variable). Moreover the experience shows that the projected points are frequently very close, which implies that rational numbers with large numerator and denominator may needed to separate projected points.

Therefore it is much better to not compute the projections explicitly. This implies that, when applying the preceding operations to a projected point, one has more variables than if one would have with an explicitly computed projection. Otherwise nothing is changed. This is applied in next section to decide to which cell of a CAD belongs a given point.

The functions described above cover almost all the needs for point geometry. We emphasize again that, when precision is not critical, one may use interval arithmetic for most computations. Therefore our representation of the points could be the best compromise between exactness (or robustness) and speed of computation.

3 Application to Cylindrical Algebraic Decomposition

Cylindrical algebraic decomposition (CAD) is a an algorithm introduced by G. Collins in 1975 [3] to implement Tarski quantifier elimination. Theoretically it may solve every problem of real

algebraic geometry which may be expressed by a formula of first order logic. Its large generality makes it inefficient for many problems. Nevertheless it is among the most important algorithms in geometry. It is also the most known algorithm which systematically uses points with algebraic coordinates.

In this section, we show how above representation of the points may be used to compute a CAD. For being understandable, it is useful to sketch a description of the algorithm.

The input of CAD is a set $\mathcal{A}_n$ of polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ and the output is a decomposition of the space $\mathbb{R}^n$ into connected sets called *cells* on which the polynomials in $\mathcal{A}_n$ have a constant sign $+$, $-$ or 0. The algorithm splits into two steps respectively called *projection* and *lifting*.

During the projection step a finite set $\mathcal{A}_i \subset \mathbb{Q}[x_1, \ldots, x_i]$ is recursively computed for $i = n-1, \ldots, 1$. These $\mathcal{A}_i$'s, called *projection sets*, are computed by purely rational algorithms, like resultants. Their computation is thus not interesting here.

The elements in $\mathcal{A}_1$ define a partition of $\mathbb{R}$ in their real roots and the intervals between them. The lifting step consists, for every cell in $\mathcal{A}_i$ to partition the cylinder above it in $\mathbb{R}^{i+1}$ into *sections* and *sectors*, which are the cells in $\mathbb{R}^{i+1}$. A section is a root of some polynomial in $\mathcal{A}_{i+1}$; more precisely, the definition of $\mathcal{A}_i$ implies that the zeros above any cell of the polynomials in $\mathcal{A}_{i+1}$ consist in a finite number of regular sheets which do not cross. The sectors are the parts delimited by the sections of the cylinder above the cell under consideration.

In classical CAD, the cells are represented by a sample point and by the numbering of the roots which define it. A semi-algebraic description of the cells may also be provided, but it may dramatically increase the number of cells when polynomial of high degree are involved. In our version, the cells are simply represented by their sample point and the list of variables for which they are sections or sectors. A semi-algebraic description of the cells is not immediately available. As such a description it is only useful, in practice, to decide to which cell belongs a given point, we replace it by an algorithm for this decision.

With our representation of the points, the algorithm to lift to $\mathbb{R}^{i+1}$ a cell in $\mathbb{R}^i$ represented by a sample point p is the following.

• *For $f \in \mathcal{A}_{i+1}$ convert to points* RSOLVE(EQS(p) $\cup \{f\}$, INEQ(p)). *If the output of* RSOLVE *is "non zero-dimensional system" this results in an empty set of points.*

• *Sort all these points (corresponding to all the f's) by the values of x_{i+1}. For this, one may take advantage that the equality of two values of x_{i+1} implies the equality of the corresponding points (see preceding section). Output as sections the resulting set of points.*

• *Choose a rational number r in each interval defined by the x_{i+1} coordinates of these sections and output as sector the point q defined by* EQS(q)$= \{x_{i+1} - r\} \cup$ EQS(p), INEQ(q) $=$ INEQ(p) *and* APPROX(q) $= \{x_{i+1} = [r, r]\} \cup$ APPROX(p).

It clear that this algorithm is more efficient if one suppose the polynomials in $\mathcal{A}_{i+1}$ irreducible: RSOLVE is called with polynomials of smaller degrees and it is easier to separate the roots. On the other hand, factoring is not very costly in the context of CAD.

To decide to which cell belongs a given point p, we proceed by induction on i, the dimension of the space on which the configuration is projected. To start this induction, we have to compare the first coordinate of p with the roots of the polynomials in $\mathcal{A}_1$. This is done by the algorithm of preceding section and decides to which cell belongs the projection of p in $\mathbb{R}^1$.

Now, suppose, by induction hypothesis, that the projection of p into $\mathbb{R}^i$ belongs to a cell in $\mathbb{R}^i$ whose sample point is q. The comparison of the x_{i+1} coordinate of p with the values of the sections above p decides in which cell of $\mathbb{R}^{i+1}$ belongs the projection of p on this space. In fact, the arrangement of the sections is the same above q and the projection of p into $\mathbb{R}^i$.

We have thus to compute the points which are zeros of a polynomial $f \in \mathcal{A}_{i+1}$ and have the same projection as p into $\mathbb{R}^i$. When these points are known, the above described comparisons are

straightforward using the algorithms of the preceding section. To compute these points, let p' be the point obtained by substituting x_j by a new variable y_j in p, for $j = i+1, \ldots, n$. Then the desired points are obtained by converting to points the output of RSOLVE($\{f\} \cup$ EQS(p'), INEQ(p')). The fact that the desired points involve more variables than $x_1, \ldots, x_{i+1}$ does cause any trouble.

4 An example

A challenging example of quantifier elimination appears in [4]. It consists in writing the condition on a, b, c, d for an ellipse of equation $\frac{(x-c)^2}{a} + \frac{(x-d)^2}{b} - 1 = 0$ (with $a > 0$ and $b > 0$) being contained in the disk of equation $x^2 + y^2 < 1$.

This is challenging problem because, although it has been solved in 1988 [4], it was impossible, until recently, to get an automatic solution through CAD. The projection step of the CAD has been completed by C. Brown in 1981 [1].

Let P be the discriminant with respect to y of the resultant with respect to x of the two equations. It is easy to show that the ellipse is inside the disk if and only if the circle and the ellipse do not intersect in the real space and the ellipse has a point inside the disk. By a result in [2], this implies that, to solve this problem of quantifier elimination, it suffices to compute a CAD of P and to test, for each sample point, if above condition is fulfilled.[†]

We have run on our laptop an experimental version (written in Maple) of our way of implementing CAD on this polynomial P. The CAD computation took 37 seconds, essentially devoted to lifting process and produced 1467 cells. Making the list of the 231 cells for which the ellipse is inside the disk took 8.5 minutes more (1467 polynomial systems with 6 variables to solve). On the other hand, C. Brown has run QEPCADB on the same problem. The CAD computation took 53 sec. The description of the resulting cells do not allow to decide easily for which cells the circle and the ellipse do not intersect.

This comparison is not really significant, as the computations have been done on different computers, with very different software and different optimizations. However, it shows at least that our method is promising and competitive, and allows further computation with the output, which otherwise are not easy.

References and Notes

[1] Christopher W. Brown. Improved projection for cylindrical algebraic decomposition. *J. Symbolic Comput.*, 32(5):447–465, 2001.

[2] Christopher W. Brown and Scott McCallum. On using bi-equational constraints in CAD construction. In *ISSAC'05*, pages 76–83 (electronic). ACM, New York, 2005.

[3] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata theory and formal languages (Second GI Conf., Kaiserslautern, 1975)*, pages 134–183. Lecture Notes in Comput. Sci., Vol. 33. Springer, Berlin, 1975.

[4] Daniel Lazard. Quantifier elimination: optimal solution for two classical examples. *J. Symbolic Comput.*, 5(1-2):261–266, 1988.

[†] Instead of P the authors of [2] consider a multivariate discriminant which differs from P only by factors $a - b$, a, b, c. It appears that these factors occur, in any case, in the $\mathcal{A}_i$ and the CAD's of P and of the discriminant of [2] are thus the same.

On Using Triangular Decomposition for Solving Parametric Polynomial Systems

Fabrice Rouillier[1], Rong Xiao[1,2]

[1] SALSA INRIA project-team, INRIA Paris-Rocquencourt & Laboratoire d'Informatique de Paris VI, France
Fabrice.Rouillier@inria.fr

[2] School of Mathematical Sciences, Peking University Beijing, China
akelux@gmail.com

1 Introduction and Problem

Solving a polynomial system with parametric coefficients theoretically reduces (at least) to compute a (the minimal) discriminant variety. That is to understand the geometric properties of the following canonical projection map Π_{U}

$$\Pi_{\mathrm{U}} : \mathcal{C} \subset \mathbb{C}^{s+d} \mapsto \mathbb{C}^d$$
$$\Pi_{\mathrm{U}}(x_1, \cdots, x_s, u_1, \cdots, u_d) = (u_1, \cdots, u_d).$$

1.1 Discriminant variety

In [4], the authors proved that the complement of the union of all open subsets in $\overline{\Pi_{\mathrm{U}}(\mathcal{C})}$ with $(\Pi_{\mathrm{U}}, \mathcal{C})$-covering property is a variety, and called it *the minimal discriminant variety*, denoted by W_D. They also proved that W_D is in fact the union of following sets which intuitively define "non-generic" parameter values:

- O_{sd}, the projection of *small dimensional components*;
- O_{∞}, the set of non properness of Π_{U} restrict on $\mathcal{C}$;
- O_{sing}, the singular points of $\overline{\Pi_{\mathrm{U}}(\mathcal{C})}$;
- O_F, the projection of the intersection of $\overline{\mathcal{C}}$ with hypersurface $\Pi_{i=1}^{k} g_i = 0$.

Computing the minimal discriminant variety intrinsic object is harder than to compute a larger discriminant variety, which includes the minimal one but with similar properties.

A large class of parametric systems are so called *well-behaved systems*, where there are as many equations as unknowns and specialized to 0-dimensional and radical system for almost all parameters' values. For this kind of systems, $W_{sd} = W_{sing} = \emptyset$, and in [4], the authors proposed an optimized/efficient algorithm to compute the minimal DV.

Besides well-behaved systems, there are lots of *non well-behaved systems* also. For such kind of systems, major computational difficulties of computing even a large DV occur in:

- the computation of the singular and critical loci;
- the projection of the components of dimension less than d;

Other side effects such as the presence of embedded components may also disturb the computation or at least prevent from computing the minimal discriminant variety.

*Correspondence to: Fabrice.Rouillier@inria.fr, akelux@gmail.com

1.2 Motivations and contributions

One solution to deal with non-well-behaved systems is to decompose first the system to (at least) equi-dimensional and radical components, some practical decompositions or comparisons can be find in [2, 5]. We thus focus on the use of triangular decomposition to solve parametric system with following main objectives:

- to compute DV for general parametric systems;
- to measure the difference between computed large DV and the minimal DV.
- propose optimizations to speed up triangular decomposition adaptive to discriminant variety computation.

For triangular systems, we find the correspondence (inclusion or equal) relation between the intrinsic DV components and the natural parameter's subsets vanishing on initials, separants, or inequation polynomials. We show that these remarkable subsets form the minimal discriminant variety of a triangular system. For general systems, we study some proper but most general specifications of triangular decomposition and constructing DV therefrom. We also discover that the computed DV is not minimal in general and analysis the redundancy. We also propose several techniques to optimize triangular decompositions for decomposing and solving parametric system more efficiently.

2 Computing DV from Triangular Decomposition

To simplify the discussion, we assume that $\overline{\Pi_{\mathrm{U}}(C)}$ is the full parametric space, and Π_{U} generically has finite fibers. We will take a component view in parametric system solving. Let C be an irreducible component of variety $\mathbf{V}(\mathcal{C})$. Then C is called a *basic main component* of $\mathcal{C}$ if $\dim(\Pi_{\mathrm{U}}(C)) = \dim(\Pi_{\mathrm{U}}(\mathcal{C}))$; otherwise C is called a *basic suspicious component*.

The system defined as bellow is the most general triangular system we expect from triangular decomposition.

Definition 1 *A polynomial system* $[\mathbf{T}, \mathbf{Q}]$ *is a* regular separable triangular system *(R.S.TS.) if* $\mathbf{T}$ *is a square free regular chain [1, 3] and* $\forall q \in \mathbf{Q}$, q *is regular w.r.t.* $\mathbf{T}$.

Note that an R.S.TS. is similar to a simple system [6] or a regular system [2] with square free regular chains defining quasi-component, but differs by some details. In fact, R.S.TS. is less restrictive compared to simple systems or regular systems defining quasi-components. Thus R.S.TS. triangular decomposition should be more easy to compute in theory.

2.1 Basic case

Suppose that $\mathbf{T} = \{f_1, f_2, \cdots, f_s\}$, and $\mathcal{C} = [\mathbf{T}, \mathbf{Q}]$ is a well-behaved parametric R.S.TS.. The following notations are introduced to denote some natural objects related to the triangular structure of $\mathcal{C}$.

- B_{ini} denotes the product of $\mathrm{ini}(f_1)$ and $\mathrm{ires}(\mathrm{ini}(f_{i+1}), \mathbf{T})$, $\forall i = 1, \cdots, s-1$;
- B_{sep} denotes the product of $\mathrm{ires}(\mathrm{discrim}(f_i, \mathrm{mv}(f_i)), \mathbf{T})$, $\forall i = 1, \cdots, s$.
- B_{ie} denotes the product of $\mathrm{ires}(q, \mathbf{T})$ $\forall q \in \mathbf{Q}$ if $\mathbf{Q}$ is not empty, else 1.

$\mathbf{H}_{ini}$, $\mathbf{H}_{sep}$ and $\mathbf{H}_{ie}$ will denote the hypersurfaces in the parametric space defined by $B_{ini} = 0$, $B_{sep} = 0$, and $B_{ie} = 0$ respectively.

The product of B_{ini}, B_{sep}, B_{ie} is called a *border polynomial* of parametric system$[\mathbf{T}, \mathbf{Q}]$ by Yang and Xia in [7] for real root classification. Thought B_{ini}, B_{sep}, B_{ie} are algorithmically defined, they are close linked to intrinsic non-generic parametric sets, as in following theorem.

Theorem 1 *Let $\mathcal{C} = [\mathbf{T}, \mathbf{Q}]$ be a well-behaved parametric R.S.TS.. Then*

- *the projection of all suspicious components of $\mathcal{C}$ is included in $\mathbf{H}_{ini}$;*
- *$W_\infty(\mathcal{C}) \subset \mathbf{H}_{ini}(\mathbf{T})$ and if $\mathbf{V}(\mathcal{C}) = \mathbf{V}(\mathbf{T}$, e.g. $\mathbf{Q} = \emptyset$, then $W_\infty(\mathcal{C}) = \mathbf{H}_{ini}(\mathbf{T})$;*
- *$W_c(\mathcal{C}) \subset \mathbf{H}_{sep}(\mathbf{T})$;*
- *$W_F(\mathcal{C}) \subset \mathbf{H}_{ie}(\mathbf{T})$.*

So $\mathbf{H}_{ini} \cup \mathbf{H}_{sep} \cup \mathbf{H}_{ie}$ is a DV of $\mathcal{C}$, since W_{sd}, W_{sing} are trivially empty. Indeed, $\mathbf{H}_{ini} \cup \mathbf{H}_{sep} \cup \mathbf{H}_{ie}$ is minimal.

Theorem 2 *Let $\mathcal{C} = [\mathbf{T}, \mathbf{Q}]$ be a well-behaved R.S.TS.. then $\mathbf{H}_{sep} \cup \mathbf{H}_{ini} \cup \mathbf{H}_{ie}$ is the minimal discriminant variety of it.*

2.2 General case

Suppose that $\mathcal{C} = [\mathbf{P}, \mathbf{Q}]$, $\mathfrak{D}$ is an R-irredundant decomposition of $\mathcal{C}$, $[\mathbf{T_m}, \mathbf{Q_m}]$ is a well-behaved subsystem in $\mathfrak{D}$. The *revised discriminant variety* of $[\mathbf{T_m}, \mathbf{Q_m}]$ w.r.t. $\mathcal{C}$ is defined to be the minimal DV of $[\mathbf{T_m}, \mathbf{Q_m} \cup \mathbf{Q}]$. The following theorem shows how to construct a DV from an R-irredundant R.S.TS. decomposition.

Theorem 3 *Suppose $\mathcal{C} = [\mathbf{P}, \mathbf{Q}]$, $\mathfrak{D}$ is an R-irredundant decomposition of $\mathcal{C}$. Then the union of revised* DV*s of all well-behaved subsystems in $\mathfrak{D}$, projection of pairwise intersections of all well-behaved subsystems, and projection of non well behaved subsystems, is a* DV *of $\mathcal{C}$.*

2.3 Minimality and redundancies

For general systems, the DVs constructed from triangular decomposition as in Theorem 3 are not minimal in general.

Basically, the redundancy come from two aspects: one is that the regular chain $\mathbf{T}$ may not be "reduced" when representing the ideal $\mathrm{sat}(\mathbf{T})$, this redundancy can be eliminated by compute a canonical form of $\mathbf{T}$, but still can not ensure $\mathbf{H}_{ini}(\mathbf{T}) = O_\infty(\mathrm{sat}(\mathbf{T}))$; another redundancy comes from the case when there is a regular chain $\mathbf{T}_s$ with lower dimensional, but $\mathrm{sat}(\mathbf{T}_s) \supset \mathrm{sat}(\mathbf{T})$.

3 Speed up techniques

3.1 Computing over parameters' function field

Getting main components is very important for solving parametric systems. We will show that one can compute main components by carrying out a triangular decomposition over the function field $\mathbb{Q}(U)$, which is more easy/efficient to compute compared to other strategies according to our experiments.

Theorem 4 *Given a parametric system $\mathcal{C}$ with parameters* U *and unknowns* X*, suppose that* $\mathbf{T}_i$ $(i = 1, \cdots, o)$ *is an R-irredundant decomposition w.r.t.* $\mathrm{ord}_{\mathrm{X}}$ *of* $\mathcal{C}$ *as a* 0*-dimensional constructible set of* X *with coefficients from* $\mathbb{Q}[U]$ *s.t.*$\mathbf{T}_i \subset \mathbb{Q}[X, U]$*. Then viewing* $\mathbf{T}_i$ $(i = 1, \cdots, o)$ *as regular chains in* $\mathbb{Q}[X, U]$*,* $\cup_{i=1}^{o} \mathbf{V}(\mathrm{sat}(\mathbf{T}_i)$ *is the main component of* $\mathcal{C}$*.*

3.2 Lazy decomposition

A full triangular decomposition contains all the sufficient information to compute a discriminant variety therefrom, however it overdoes to decompose everything and make the components triangular, but thus blocks often, in practice, the decomposition. By computing a triangular decomposition over the function field of parameters, one can get directly the main components efficiently but can not track all information about suspicious components.

For parametric systems, decomposing suspicious components whose projection is inside the minimal DV is useless, and most of the time, time consuming. So we should avoid such decomposition as much as we can, which we call a *lazy decomposition*.

4 Conclusion

A first conclusion is that decomposing a system into triangular sets is a competitive way for solving general parametric systems, but a second part of the conclusion is that, up to our knowledge of the current state of the art, there dos not currently exists any "universal" efficient method. However, mix different strategies can solve more complicated problems. Our further work will focus on developing more adaptive algorithms for decomposing parametric systems, integrating various speed up strategies for different purposes and at different levels, mixing with Gröbner based strategies with an ultimate goal of providing efficient solutions for computing (the minimal) DV in the general case (non well-behaved systems).

References and Notes

[1] Philippe Aubry, Daniel Lazard, and Marc Moreno Maza. On the theories of triangular sets. *J. Symb. Comput.*, 28(1-2):105–124, 1999.

[2] Changbo Chen, Oleg Golubitsky, Franois Lemaire, Marc Moreno Maza, and Wei Pan. Comprehensive triangular decomposition. In *Proceedings of CASC 2007*, pages 73–101, 2007.

[3] M. Kalkbrener. A generalized euclidean algorithm for computing triangular representations of algebraic varieties. *Journal of Symbolic Computation*, 15(2):143–167, 1993.

[4] Daniel Lazard and Fabrice Rouillier. Solving parametric polynomial systems. *J. Symb. Comput.*, 42(6):636–667, 2007.

[5] Guillaume Moroz. Regular decompositions. In *Proceedings of ASCM 2007*, pages 263–277, Singapore, 2007.

[6] Dongming Wang. *Elimination Methods*. Springer-Verlag, Wien New York, 2001.

[7] Lu Yang, Xiaorong Hou, and Bican Xia. Automated discovering and proving for geometric inequalities. In *Proceedings of ADG 1998*, pages 30–46, London, UK, 1998.

A Sum of Squares Approach to Nonlinear Gain Analysis of a Class of Nonlinear Dynamical Systems

Hiroyuki Ichihara[1*] and Hirokazu Anai[2,3]

[1] Department of Systems Design and Informatics, Kyushu Institute of Technology, Iizuka, Japan
ichihara@ces.kyutech.ac.jp

[2] Design Innovation Laboratory, Fujitsu Laboratories Ltd., Kanagawa, Japan
[3] Graduate School of Mathematics, Kyushu University, Fukuoka, Japan
h.anai@kyudai.jp

Abstract

A numerically tractable condition is proposed for input-to-state stability (ISS) analysis for a class of nonlinear systems in terms of sum of squares (SOS). The concept of ISS has been explored in nonlinear control theory while computational approaches including numeric and symbolic have been less attention. If class of the systems is restricted to polynomial ones, then it is possible to use SOS relaxation for ISS analysis. In this paper, a numerical method for nonlinear gain function is proposed by using SOS relaxation and semidefinite programming. A numerical example is shown to illustrate the proposed method in which some infeasibility check is performed by a symbolic method, quantifier elimination.

1 Introduction

Input-to-state stability (ISS) is a standard concept of stability of nonlinear dynamical systems [1, 2]. The notion of ISS was first introduced by Sontag [3], which involves requirements on zero-state response and zero-input response by a nonlinear gain function. An important equivalent condition to ISS has been proposed using the ISS Lyapunov function [4]. On the other hand, analysis and synthesis of nonlinear systems have been developed by a numerical method, sum of squares (SOS)[5, 6] and semidefinite programming. In general, such relaxation method can treat a several engineering problems but generates a conservative result. To overcome the drawback, symbolic approaches should be paid attention.

In this paper, we give a SOS condition for ISS analysis. The condition originates from a dissipation type ISS inequality. The nonlinear gain function can be constructed from a feasible solution satisfying the condition. One of the keys of this formulations is to realize class $\mathcal{K}_\infty$ functions, which has inverse map on real nonnegative region, by polynomials. A simple example is shown to illustrate the ISS stability analysis. Since the SOS approach can not make decision exactly whether there exists a linear gain, we use a symbolic approach to do this task.

Notation: $\|\cdot\|$ represents the Euclidean norm. $\Sigma(x)$ represents the class of SOS polynomials of variable vector $x \in \mathbb{R}^n$.

*680-4, Kawazu, Iizuka, 820-8502, Japan, Tel:+81-948-29-7748, Fax:+81-948-29-7709

2 Nonlinear Gain Analysis

Consider the system

$$\dot{x}(t) = f(x(t), w(t)),\ x(0) = x_0,\ t \geq 0 \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state, $w(t) \in \mathbb{R}^p$ is an exogenous input, $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is a continuously differentiable function. Suppose that $f(0,0) = 0$ and $w(t)$ is a piecewise continuous, bounded function in t for all $t \geq 0$.

Definition 1 ([4]) *The system (1) is said to be input-to-state stable if there exists a class $\mathcal{KL}$ function β and a class $\mathcal{K}$ function γ, called a nonlinear gain function, such that for any initial state $x(0)$ and any bounded input $w(t)$, the solution $x(t)$ exists for all $t \geq 0$ and satisfies*

$$\|x(t)\| \leq \beta(\|x_0\|, t) + \gamma\Big(\sup_{\tau \in [0,t]} \|w(\tau)\|\Big). \tag{2}$$

Definition 2 ([4]) *A continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ is called a ISS-Lyapunov function for the system (1) if there exist class $\mathcal{K}_\infty$ functions α_i, $i = 1, 2, 3$, a class $\mathcal{K}$ function ρ such that*

$$\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|),\ \forall x \in \mathbb{R}^n \tag{3}$$

$$\frac{\partial V}{\partial x} f(x, w) \leq -\alpha_3(\|x\|),\ \forall \|x\| \geq \rho(\|w\|). \tag{4}$$

Then $\gamma(r) = \alpha_1^{-1} \circ \alpha_2 \circ \rho(r)$.

Lemma 1 ([4]) *The system (1) is input-to-state stable with γ if and only if there exist a continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$, class $\mathcal{K}_\infty$ functions α_i, $i = 1, \ldots, 3$, class $\mathcal{K}$ function σ such that (3) and*

$$\frac{\partial V}{\partial x} f(x, w) \leq \sigma(\|w\|) - \alpha_3(\|x\|) \tag{5}$$

for all $(x, w) \in \mathbb{R}^n \times \mathbb{R}^p$. Then $\gamma(r) = \alpha_1^{-1} \circ \alpha_2 \circ \alpha_3^{-1} \circ c\ \sigma(r)$ for some $c > 1$.

3 SOS Conditions for ISS Analysis

We assume that f in the system (1) is a polynomial. To rewrite the conditions in Lemma 1 as some numerically solvable one, it is required that class $\mathcal{K}$ and $\mathcal{K}_\infty$ functions in the lemma are realized by polynomials.

Lemma 2 *Consider a univariate real even polynomial without constant term*

$$\alpha(r) = \textstyle\sum_{\kappa=1}^{N} c_\kappa r^{2\kappa}. \tag{6}$$

Then $\alpha(r)$ belongs to class $\mathcal{K}_\infty$ if and only if there exist scalars c_κ, a SOS polynomial $s_{\alpha 0}(r)$ such that

$$r \cdot d\alpha(r)/dr = s_{\alpha 0}(r),\ \forall r \in \mathbb{R}. \tag{7}$$

Theorem 1 *The system (1) is input-to-state stable with γ if there exist even polynomials $\alpha_i \in \mathcal{K}_\infty$, $i = 1, \ldots, 4$, a polynomial $V(x)$, sum of squares polynomials $s_{10}(x)$, $s_{20}(x)$ and $s_{30}(x, w)$ such that*

$$V(x) - \alpha_1(\|x\|) = s_{10}(x), \tag{8}$$
$$\alpha_2(\|x\|) - V(x) = s_{20}(x), \tag{9}$$
$$\frac{\partial V}{\partial x} f(x, w) - \alpha_4(\|w\|) + \alpha_3(\|x\|) = -s_{30}(x, w) \tag{10}$$

for all $(x, w) \in \mathbb{R}^n \times \mathbb{R}^p$. Then $\gamma(r) = \alpha_1^{-1} \circ \alpha_2 \circ \alpha_3^{-1} \circ c\ \alpha_4(r)$ for some $c > 1$.

Remark 1 *One can set $\alpha_3(\|x\|) = \gamma\|x\|^2$ and $\alpha_4(\|w\|) = \|w\|^2$ with $\gamma > 0$ in Theorem 1. If there exists such γ, then it is a linear gain. However, it is not always true that there does not exists a linear gain when the SOS problem is infeasible. The reason is a gap between positive polynomials and SOS polynomials. To check the feasibility of the problem exactly, we can use quantifier elimination (QE)[9].*

4 Symbolical and Numerical Example

Consider

$$\dot{x} = -x^3 + (x^2 + 1)w.$$

We choose a candidate Lyapunov function as $V(x) = x^4/4 \in \mathcal{K}_\infty$. First we try to search a liner gain for this system. That is, we would like to decide the feasibility of the problem

$$\exists \gamma (> 0) \text{ s.t. } x^3 \left\{-x^3 + (x^2 + 1)w\right\} - \|w\|^2 + \gamma\|x\|^2 \leq 0\ \forall x\ \forall w.$$

Performing QE to this problem, we obtain the result "*false*"and thus the problem is infeasible for $V(x)$. Next, we take the SOS approach by Theorem 1, in which we set $\alpha_3(r)$ and $\alpha_4(r)$ as

$$\alpha_3(r) = b_6 r^6, \quad \alpha_4(r) = c_2 r^2 + c_4 r^4 + c_6 r^6.$$

To solve the SOS problem, we use SOS parser YALMIP [10] and SDP solver SeDuMi [11]. By using the result in Theorem 1, our problem is

$$\begin{aligned}
&\exists \alpha_3, \alpha_4 \in \mathcal{K}_\infty, s_{30} \in \Sigma(x, w), s_{50} \in \Sigma(w), b_6, c_2, c_4, c_6 \in \mathbb{R} \\
&\text{s.t. } x^3 \left\{-x^3 + (x^2 + 1)w\right\} - \alpha_4(\|w\|) + \alpha_3(\|x\|) = -s_{30}(x, w)\ \forall x\ \forall w, \\
&\quad b_6 > 0, \\
&\quad (c_2 w + 2c_4 w^3 + 3c_6 w^5)w = s_{50}(w)\ \forall w.
\end{aligned}$$

This SOS problem is feasible and a solution is $\alpha_3(r) = 0.13085 r^6$ and $\alpha_4(r) = 1.8654 r^2 + 1.3828 r^4 + 1.1448 r^6$,

$$\begin{aligned}
S_{30} = &\begin{bmatrix} w^2 \\ xw \end{bmatrix}^T \begin{bmatrix} 1.1422 & -0.36102 \\ -0.36102 & 0.70818 \end{bmatrix} \begin{bmatrix} w^2 \\ xw \end{bmatrix} \\
&+ \begin{bmatrix} w \\ w^3 \\ xw^2 \\ x^2 w \\ x^3 \end{bmatrix}^T \begin{bmatrix} 1.8654 & 0.12031 & 0.36102 & -0.35409 & -0.5 \\ 0.12031 & 1.1448 & 0.0 & -0.63304 & 0.039198 \\ 0.36102 & 0.0 & 1.2661 & -0.039198 & -0.6655 \\ -0.35409 & -0.63304 & -0.039198 & 1.331 & -0.5 \\ -0.5 & 0.039198 & -0.6655 & -0.5 & 0.86915 \end{bmatrix} \begin{bmatrix} w \\ w^3 \\ xw^2 \\ x^2 w \\ x^3 \end{bmatrix}, \\
S_{50} = &\ 3.6172 w^4 + \begin{bmatrix} w \\ w^3 \end{bmatrix}^T \begin{bmatrix} 3.7308 & 0.95699 \\ 0.95699 & 6.8687 \end{bmatrix} \begin{bmatrix} w \\ w^3 \end{bmatrix}.
\end{aligned}$$

The nonlinear gain is $\gamma(r) = \alpha_3^{-1}(c\ \alpha_4(r))$ for some $c > 1$.

5 Summary

We have obtained SOS conditions for ISS analysis problems for polynomial systems, in which class $\mathcal{K}_\infty$ functions are realized computationally by means of polynomials. The proposed SOS condition is convex so that one does not need any iterative algorithms or bilinear solvers. QE approach is performed to confirm non-existence of a linear gain.

References and Notes

[1] H. K. Khalil, Nonlinear Systems, 3rd edition, Prentice Hall, 2001

[2] A. Isidori, Nonlinear Control Systems II, Springer, 1999

[3] E. D. Sontag, Smooth stabilization implies coprime factorization, IEEE Transaction on Automatic Control, Vol.34, No.4, 435-443 (1989)

[4] E. D. Sontag and Y. Wang, On characterizations of the input-to-state stability property, Systems & Control Letters, Vol.24, No.4, 351-359 (1995)

[5] P. A. Parrilo, Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization, California Institute of Technology, Pasadena, CA (2000)

[6] Z. Jarvis-Wloszek and R. Feeley and W. Tan and K. Sun and A. Packard, Some Controls Applications of Sum of Squares Programming, Proc. IEEE Conf. on Decision and Control, 4676–4681 (2003)

[7] Z. P. Jiang and A. R. Teel and L. Praly, Small-gain theorem for ISS systems and applications, Mathematics of Control, Signals and Systems, Vol.7, 95-120 (1994)

[8] A. R. Teel, A nonlinear small gain theorem for the analysis of control systems with saturation, IEEE Transaction on Automatic Control, Vol.41, 1256-1270 (1996)

[9] B. Caviness and J. Johnson, editors. Quantifier Elimination and Cylindrical Algebraic Decomposition, Texts and monographs in symbolic computation, Springer-Verlag (1998)

[10] "J. Löfberg", "YALMIP : A Toolbox for Modeling and Optimization in MATLAB", "Proceedings of the CACSD Conf.", "http://control.ee.ethz.ch/~joloef/yalmip.php" (2004)

[11] J. F. Sturm,SeDuMi: Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones (updated for version 1.05), http://sedumi.mcmaster.ca/ (1998)

Note: Definition of class $\mathcal{K}_\infty$ function

Definition 3 ([1]) *A continuous function α: $[0, a) \longrightarrow [0, \infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$. It is said to belong to class $\mathcal{K}_\infty$ if $a = \infty$ and $\alpha(r) \longrightarrow \infty$ as $r \longrightarrow \infty$.*

On the computation of the optimal H_∞ norm of a parametric system achievable by a feedback controller

Takuya Kitamoto[1*] and Tetsu Yamaguchi[2]

[1] Faculty of Education, Yamaguchi University, 1-1677 Yoshida, Yamaguchi, Japan
kitamoto@yamaguchi-u.ac.jp

[2] Cybernet Systems, Co, LTD, Tokyo, Japan
tetsuy@cybernet.co.jp

Abstract

When a given system contains unknown parameters, it is natural to deal with the parameters symbolically. However, numerical computations, instead of symbolic ones, have been used for the design/analysis of a control system, due to the heavy computational complexities of symbolic computations. Recently, the computational power of a personal computer is increasing rapidly, which makes applications of symbolic computations more practical. In [4] and [5], the current authors proposed an algorithm to compute the optimal H_∞ norm of a given parametric system achievable by a static/output feedback controller. The algorithm computes the optimal H_∞ norm as a root of a polynomial. In other words, the algorithm computes the defining polynomial of the optimal H_∞ norm. As is remarked in the conclusion of [5], one drawback of the algorithm is its heavy computational complexities, and more efficient algorithms were left as future works. This paper presents a more efficient alternative to the algorithm in [5]. The key for the algorithm is the techniques used in [6] and [7]. Utilizing the techniques, we modified the algorithm in [5], which speeds up the algorithm significantly.

1 Introduction

Conventionally, a control system design/analysis is often performed by numerical methods with softwares such as MATLAB or SCILAB. Computer algebra system such as Maple or *Mathematica* were not so popular as controller design tools, due to the heavy computational complexities of symbolic computations. However, recent rapid increase of computational power of personal computers has made applications of symbolic computations more practical. In fact, a lot of attempts have been made to apply symbolic computations to controller designs. For example, [1]-[3] apply Quantifier elimination (QE), which is a new technique of computer algebra, to the design of control systems. Another attempt is [4] and [5], which is the basis of this paper. In [4] and [5], given a parametric system, the current authors proposed an algorithm to compute the optimal H_∞ norm of the system achievable by a static/output feedback controller. The algorithm computes the optimal H_∞ norm as a root of a polynomial (in other words, the algorithm computes the defining polynomial of the optimal H_∞ norm). We note that, conventionally, the optimal H_∞ norm is computed with numerical methods, and those numerical methods can not be applied to a parametric system. Although the algorithm in [5] has a such advantage over a conventional numerical method, it also has a serious drawback. As is remarked in the conclusion of [5], the algorithm in [5] has quite heavy computational complexities, and more efficient algorithms were left as future works.

*Correspondence to: Yamaguchi University, 1-1677 Yoshida, Yamaguchi, Japan Tel: +81-83-933-5336

This paper presents a more efficient alternative to the algorithm in [5]. The key for the algorithm is the techniques used in [6] and [7], where efficient algorithms to compute the defining polynomial of the solution of algebraic Riccati equation ([6]) and the determinant of a generalized Vandermonde matrix ([7]) are presented.

This extended abstract is composed as follows: In Section 2, we describe basic algorithm in [5]. Then, in Section 3, we describe how to improve the efficiency of the algorithm.

2 Basic Algorithm

Suppose that we are given a plant described by the linear differential equation

$$\frac{dx}{dt} = Ax + B_1 w + B_2 u,\ z = C_1 x + D_{12} u,\ y = C_2 x + D_{21} w, \tag{1}$$

where $A, B_1, B_2, C_1, C_2, D_{12}, D_{21}$ are matrices whose entries are polynomial in parameter k (we assume that A is an $n \times n$ matrix). In [5], the following algorithm to compute polynomial $f(q,k)$ is presented:

⟨1⟩ Let matrices H_1, H_2 be

$$H_1 = \begin{bmatrix} A & qB_1B_1^T - B_2B_2^T \\ -C_1^TC_1 & -A^T \end{bmatrix},\ H_2 = \begin{bmatrix} A^T & qC_1^TC_1 - C_2^TC_2 \\ -B_1B_1^T & -A \end{bmatrix}, \tag{2}$$

and let $g_1(x), g_2(x)$ be the characteristic polynomials of H_1 and H_2, respectively.

⟨2⟩ By solving linear equations $(H_1 - \lambda I)v_1 = 0$, $(H_2 - \mu I)v_2 = 0$, compute eigenvectors

$$v_1(\lambda) = [v_{1,1}(\lambda), \cdots, v_{1,2n}(\lambda)]^T,\ v_2(\mu) = [v_{2,1}(\mu), \cdots, v_{2,2n}(\mu)]^T$$

of H_1 and H_2 corresponding to λ and μ, respectively ($v_{1,i}(\lambda)$ and $v_{2,i}(\mu)$ are computed as polynomials in λ, q, k and μ, q, k, respectively).

⟨3⟩ For i, j satisfying $1 \le i, j \le 2$, define $n \times n$ matrices $\Lambda_{i,j}(y_1, \cdots, y_n)$ by

$$\begin{bmatrix} \Lambda_{1,j}(y_1, \cdots, y_n) \\ \Lambda_{2,j}(y_1, \cdots, y_n) \end{bmatrix} = \begin{bmatrix} v_j(y_1) & \cdots & v_j(y_n) \end{bmatrix}, \tag{3}$$

and compute polynomial $\zeta_{i,j}(y_1, \cdots, y_n)$ defined by

$$\zeta_{i,j}(y_1, \cdots, y_n) = \frac{\mathrm{Det}\left(\Lambda_{i,j}(y_1, \cdots, y_n)\right)}{\prod_{l<m}(y_l - y_m)}. \tag{4}$$

⟨4⟩ For i, j satisfying $1 \le i, j \le 2$, compute $\prod_{s_l = \pm 1} \zeta_{i,j}(s_1\lambda_1, \cdots, s_n\lambda_n)$ and remove $\lambda_1, \cdots, \lambda_n$ from the expression, using the relation

$$\lambda_1^2 \cdots \lambda_n^2 = (-1)^n g_{j,0}(q,k), \cdots, \lambda_1^2 + \cdots + \lambda_n^2 = -g_{j,2n-2}(q,k),$$

and let $\xi_{i,j}(q,k)$ be the result, where $g_{j,m}(q,k)$ denotes the coefficient of x^m of $g_j(x)$.

⟨5⟩ Let $\tau(x_1, \cdots, x_n, y_1, \cdots, y_n)$ be

$$\tau(x_1, \cdots, x_n, y_1, \cdots, y_n) = \frac{\mathrm{Det}\left(q\Lambda_{21}^T(x_1, \cdots, x_n)\Lambda_{22}(y_1, \cdots, y_n) - \Lambda_{11}^T(x_1, \cdots, x_n)\Lambda_{12}(y_1, \cdots, y_n)\right)}{\prod_{l<m}(x_l - x_m)\prod_{l<m}(y_l - y_m)}, \quad (5)$$

and compute $\prod_{t_p=\pm 1}\prod_{s_r=\pm 1}\tau(s_1\lambda_1, \cdots, s_n\lambda_n, t_1\mu_1, \cdots, t_n\mu_n)$. Then remove λ_1, $\cdots, \lambda_n, \mu_1, \cdots, \mu_n$ from the expression, using the relation

$$\begin{cases} \lambda_1^2 \cdots \lambda_n^2 = (-1)^n g_{1,0}(q,k), \cdots, \lambda_1^2 + \cdots + \lambda_n^2 = -g_{1,2n-2}(q,k), \\ \mu_1^2 \cdots \mu_n^2 = (-1)^n g_{2,0}(q,k), \cdots, \mu_1^2 + \cdots + \mu_n^2 = -g_{2,2n-2}(q,k), \end{cases} \quad (6)$$

and let $\chi(q,k)$ be the result, where $g_{j,m}(q,k)$ denotes the coefficient of x^m of $g_j(x)$.

⟨6⟩ Let $f(q,k)$ be square-free part of

$$\chi(q,k)\prod_{1\le i\le 2}\mathrm{Res}_x(g_i(x), g_i'(x))\prod_{1\le i,j\le 2}\xi_{i,j}(q,k) \quad (7)$$

and output $f(q,k)$.

Reference [5] shows that for any parameter value $k \in \Omega$ (Ω denotes valid range of the parameter values), there exists a real root α of $f(q,k)$ with respect to q such that $1/\sqrt{\alpha} = \min\|G(s)\|_\infty$, where $\min\|G(s)\|_\infty$ denotes the optimal H_∞ norm from w to z of system (1) achievable by an output feedback controller.

3 Algorithm improvement

The techniques in [6] and [7] can be used to make the above algorithm more efficient in the following way:

(i) Since matrix $\Lambda_{i,j}(y_1, \cdots, y_m)$ in (3) has a special form, the method in [7] computes $\zeta_{i,j}(y_1, \cdots, y_n)$ efficiently as a polynomial in $\sigma_1, \cdots, \sigma_n$ where $\sigma_1, \cdots, \sigma_n$ are fundamental symmetric polynomials of $y_1, \cdots, y_n$, i.e. $\sigma_1 = y_1 + \cdots + y_n, \cdots, \sigma_n = y_1 \cdots y_n$ (note that $\zeta_{i,j}(y_1, \cdots, y_n)$ is a symmetric polynomial of $y_1, \cdots, y_n$). By $\hat{\zeta}_{i,j}(\sigma_1, \ldots, \sigma_n)$, we denote the polynomial in $\sigma_1, \cdots, \sigma_n$ computed (obviously, we have $\hat{\zeta}_{i,j}(\sigma_1, \ldots, \sigma_n) = \zeta_{i,j}(y_1, \cdots, y_n)$).

(ii) Using technique in [6], $\xi_{i,j}(q,k)$ in step ⟨4⟩ can be computed as follows: Let k_0 and q_0 be some integers, and compute Groebner basis of the system

$$\{\eta_n(\sigma_1, \cdots, \sigma_n) - (-1)^n g_{j,0}(q_0, k_0), \cdots, \eta_1(\sigma_1, \cdots, \sigma_n) + g_{j,2n-2}(q_0, k_0)\} \quad (8)$$

with lexicographic ordering $\sigma_n \succ \cdots \succ \sigma_1$ where $\eta_i(\sigma_1, \cdots, \sigma_n)$ $(i = 1, \cdots, n)$ are polynomials in $\sigma_1, \cdots, \sigma_n$ satisfying

$$\eta_n(\sigma_1, \cdots, \sigma_n) = y_1^2 \cdots y_n^2, \cdots, \eta_1(\sigma_1, \cdots, \sigma_n) = y_1^2 + \cdots + y_n^2.$$

Let Groebner basis of (8) be $\{w_{j,1}(\sigma_1), w_{j,2}(\sigma_1, \sigma_2), \cdots, w_{j,n}(\sigma_1, \sigma_n)\}$ where $w_{j,1}(\sigma_1)$ and $w_{j,r}(\sigma_1, \sigma_r)$ are in the form of

$$\begin{aligned} w_{j,1}(\sigma_1) &= \sigma_1^{2^n} + w_{j,1,2^n-1}\sigma_1^{2^n-1} + \cdots + w_{j,1,0}, \\ w_{j,r}(\sigma_1, \sigma_r) &= w_{j,r,2^n}\sigma_r + w_{j,r,2^n-1}\sigma_1^{2^n-1} + \cdots + w_{j,r,0}. \end{aligned}$$

Then it can be shown that

$$\xi_{i,j}(q_0,k_0)=\mathrm{Res}_{\sigma_1}\left(\hat{\zeta}_{i,j}\left(\sigma_1,\hat{w}_{j,2}(\sigma_1),\cdots,\hat{w}_{j,n}(\sigma_1)\right),w_{j,1}(\sigma_1)\right) \tag{9}$$

where $\hat{w}_{j,r}(\sigma_1)$ $(r=1,\cdots,n)$ is the function defined by

$$\hat{w}_{j,r}(\sigma_1)=-(w_{j,r,2^n-1}\sigma_1^{2^n-1}+\cdots+w_{j,r,0})/w_{j,r,2^n}.$$

Repeating the above procedure for sufficiently many integers q_0 and k_0, we can compute polynomial $\xi_{i,j}(q,k)$ by polynomial interpolations.

(iii) $\tau(x_1,\cdots,x_n,y_1,\cdots,y_n)$ in $\langle 5\rangle$ is a symmetric polynomial in $x_1,\cdots,x_n$ and $y_1,\cdots,y_n$, and can be written as a polynomial in $\sigma_1,\cdots,\sigma_n,\theta_1,\cdots,\theta_n$, where $\theta_1=x_1+\cdots+x_n,\cdots,\theta_n=x_1\cdots x_n$ and $\sigma_1=y_1+\cdots+y_n,\cdots,\sigma_n=y_1\cdots y_n$. In this case, the method in [7] can not be applied directly. However, it is possible to use the method recursively to compute polynomial $\hat{\tau}(\sigma_1,\cdots,\sigma_n,\theta_1,\cdots,\theta_n)$ satisfying $\hat{\tau}(\sigma_1,\cdots,\sigma_n,\theta_1,\cdots,\theta_n)=\tau(x_1,\cdots,x_n,y_1,\cdots,y_n)$ efficiently.

(iv) $\chi(q,k)$ in $\langle 5\rangle$ can also be computed by polynomial interpolations as we did in the above (ii). More concretely, we have

$$\chi(q_0,k_0)=\mathrm{Res}_{\theta_1}\left(\mathrm{Res}_{\sigma_1}\left(\hat{\tau}\left(\sigma_1,\hat{w}_{1,2}(\sigma_1),\cdots,\hat{w}_{1,n}(\sigma_1),\theta_1,\hat{w}_{2,2}(\theta_1),\cdots,\hat{w}_{2,n}(\theta_1)\right),\right.\right.$$
$$\left.\left.w_{1,1}(\sigma_1)\right),w_{2,1}(\theta_1)\right).$$

References and Notes

[1] C. Abdallah, P. Dorato, W. Yang, R. Liska and S. Steinberg, "Application of Quantifier Elimination Theory to Control System Design," *Proc. of 4th IEEE Mediterranean Symposium of Control and Automation* pp. 340–345, Maleme, Crete, 1996.

[2] H. Anai and H. Yanami, "SyNRAC: A maple-package for solving real algebraic constraints," *Proc. of CASA'2003*, P.M.A. Sloot et al. (ICCS 2003) editors, Vol. 2657 of LNCS, Springer-Verlag, 2003.

[3] P. Dorato, W. Yang and C. Abdallah, "Robust Multi-Objective Feedback Design by Quantifier Elimination," *J. Symbolic Computation*, Vol. 24, pp. 153–159, 1997.

[4] T. Kitamoto and T. Yamaguchi, "The optimal H_∞ norm of a parametric system achievable using a static feedback controller," *The IEICE Trans. Funda.*, Vol. E90-A, No. 11, pp. 2496–2509, 2007.

[5] T. Kitamoto and T. Yamaguchi, "The optimal H_∞ norm of a parametric system achievable using a output feedback controller," *The IEICE Trans. Funda.*, Vol. E91-A, No. 7, pp. 1713–1724, 2008.

[6] T. Kitamoto, "Algebraic approach to the computation of the defining polynomial of the algebraic Riccati equation," *Proc. CASC 2009*, LNCS 5743, pp. 168–179, Kobe, Japan, 2009.

[7] T. Kitamoto, "On the computation of the determinant of a generalized Vandermonde matrix," submitted to IEICE Trans. on Fundamentals.

Stability Analysis for Discrete Biological Models Using Algebraic Methods

Xiaoliang Li[2], Chenqi Mou[1,2], Wei Niu[1] and Dongming Wang[1,2]

[1] Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie – CNRS,
104 avenue du Président Kennedy, F-75016 Paris, France
Wei.Niu@lip6.fr, Dongming.Wang@lip6.fr

[2] LMIB – SKLSDE – School of Mathematics and Systems Science,
Beihang University, Beijing 100191, China
xiaoliangbuaa@gmail.com, Chenqi.Mou@gmail.com

Abstract

This paper is concerned with the stability analysis of biological networks modeled as discrete and finite dynamical systems. We show how to use algebraic methods to detect steady states for such systems and to analyze the stability of each steady state for discrete dynamical systems.

1 Introduction

There are two kinds of dynamical systems, continuous and discrete, that are widely used for the modeling of biological phenomena. For biological networks modeled as continuous dynamical systems, a general algebraic approach has been proposed in [15, 20] for the detection and analysis of stability of real equilibria. This approach has also been applied to the analysis of bifurcations and limit cycles for certain biological systems [14]. In this paper, we adapt, extend, and apply the approach for stability analysis of discrete dynamical systems. Such systems may involve less parameters, but can model more complicated biological phenomena.

For discrete dynamical systems, the time domain is on fixed discrete intervals (not the real axis $\mathbb{R}$). As in the continuous case, for most nonlinear discrete dynamical systems, it is difficult to find their analytical solutions (if such solutions exist at all), so studying the qualitative behaviors of their solutions becomes an important issue. For this study, it is necessary (but highly nontrivial) to detect the equilibria of the discrete dynamical systems and analyze the stability of each equilibrium.

We consider systems of autonomous first-order discrete difference equations of the form

$$\begin{cases} x_1(t+1) = \phi_1(\mathbf{u}, x_1(t), \ldots, x_n(t)), \\ \qquad\qquad \vdots \\ x_n(t+1) = \phi_n(\mathbf{u}, x_1(t), \ldots, x_n(t)), \end{cases} \tag{1.1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{u} = (u_1, \ldots, u_m)$ are variables and parameters respectively, and $\phi_i : \mathbb{R}^{m+n} \to \mathbb{R}$ for $i = 1, \ldots, n$. Our purpose is to detect the equilibria and analyze their stability for such systems by means of symbolic computation.

We are interested in discrete time dynamical systems with state variables and parameters over the real field $\mathbb{R}$ or a finite field $\mathbb{F}_q$. The former are often used to describe the epidemic models and have been analyzed mainly by using purely mathematical methods (see, e.g., [2]), while the latter are called *finite dynamical systems* and have been used to model a variety of biological networks, such as metabolic, gene regulatory and signal transduction networks. For such systems, most of the existing studies focus on random Boolean

networks [10]. Recently, methods based on Gröbner bases [11, 12] and SAT algorithms [18] are applied to detecting equilibria for given Boolean networks. This paper reports our current investigations on the use of algebraic methods based on triangular decomposition, Gröbner bases, and real solution classification for stability analysis of discrete time and finite dynamical systems from computational biology.

2 Discrete Dynamical Systems

We first explain how to detect the equilibria of discrete dynamical systems over $\mathbb{R}$ and analyze the stability of each equilibrium. For any given values $\bar{\mathbf{u}}$ of the parameters $\mathbf{u}$, a point $\bar{\mathbf{x}} \in \mathbb{R}^n$ is called a *steady state* (or an *equilibrium*, or a *fixed point*) of the discrete system (1.1), if it is a value of the state variables $\mathbf{x}(t)$ that is invariant under further iterations of the dynamical system, i.e. $\bar{\mathbf{x}} = \phi(\bar{\mathbf{u}}, \bar{\mathbf{x}})$.

According to this definition, we may form the following system of equations:

$$\phi_1(\mathbf{u}, \mathbf{x}) - x_1 = 0, \ldots, \phi_n(\mathbf{u}, \mathbf{x}) - x_n = 0. \tag{2.1}$$

Then the problem of determining the (number of) steady states of (1.1) is reduced to two algebraic problems about the classification of real solutions of (2.1) for $\mathbf{x}$, same as Problems 1 and 2 stated in [15, 20] for continuous dynamical systems.

For the continuous case, we may use the first method of Lyapunov with the technique of linearization to analyze the stability of each steady state and to determine conditions on the parameters for steady states to be stable. A similar procedure based on linearizing the system around the steady states and determining the conditions on the eigenvalues of the Jacobian matrix $\mathbf{J}(\mathbf{u}, \mathbf{x})$ can be performed for a discrete system. The following theorem tells us how to determine the stability of a steady state $\bar{\mathbf{x}}$ for given parametric values $\bar{\mathbf{u}}$.

Theorem 2.1 ([6]) (*a*) *If all the eigenvalues of the matrix* $\bar{\mathbf{J}} = \mathbf{J}(\bar{\mathbf{u}}, \bar{\mathbf{x}})$ *lie in the open unit disk* $|\lambda| < 1$*, then* $\bar{\mathbf{x}}$ *is asymptotically stable.*

(*b*) *If the matrix* $\bar{\mathbf{J}}$ *has at least one eigenvalue* λ_0 *outside the open unit disk, i.e.* $|\lambda_0| > 1$*, then* $\bar{\mathbf{x}}$ *is unstable.*

The eigenvalues of $\bar{\mathbf{J}}$ are the roots of its characteristic polynomial A, so the problem of stability analysis can be reduced to the problem of determining whether all the roots of A lie in the open unit disk $|\lambda| < 1$. Similar to the Routh-Hurwitz criterion which is applicable to continuous systems, three criteria are available for discrete dynamical systems: generalized Routh-Hurwitz criterion [16], Schur-Cohn criterion [8] and Jury criterion [9].

These three criteria further reduce the problem of determining whether all the eigenvalues of $\bar{\mathbf{J}}$ lie in the open unit disk to that of determining the signs of certain coefficients of A and certain determinants, or the signs of certain expressions of the coefficients. This leads to two algebraic problems for the stability analysis of system (1.1), similar to Problems 3 and 4 for continuous systems formulated in [15, 20].

As we have explained above, the problem of stability analysis of discrete dynamical systems may be reduced to four algebraic problems, which are almost the same as those for continuous dynamical systems. The differences lie in the expressions of the equation system (2.1) and the polynomials derived from the criteria. Therefore, these problems may also be solved by using the approach proposed in [15, 20].

In the process of the stability analysis of some discrete biological systems of high dimension ($n > 4$) by using the three stability criteria, the determinants or expressions (if they

are obtained after heavy computation) may be too complicated to be used in the algebraic approach we mentioned before.

3 Finite Dynamical Systems

In this section, our attention is focused mainly on the simplest class of finite dynamical systems, Boolean networks, whose form is similar to (1.1). The differences are that for Boolean networks, $\phi = (\phi_1, \ldots, \phi_n) : \{0,1\}^{m+n} \to \{0,1\}^n$, are written in terms of the Boolean operators $\vee$, $\wedge$, $\neg$; the states of the variables $\mathbf{x}$ and parameters $\mathbf{u}$ are 0 and 1.

Boolean functions can be translated into polynomial functions by using the rules $x \wedge y = xy$, $x \vee y = x + y + xy$ and $\neg x = x + 1$ (see, e.g., [12]). Then determining the (number of) steady states of a Boolean network may be reduced to studying the solutions of the equation system over $\mathbb{F}_2$ as follows:

$$g_1(\mathbf{u}, \mathbf{x}) - x_1 = 0, \ldots, g_n(\mathbf{u}, \mathbf{x}) - x_n = 0, \tag{3.1}$$

where g_i is the polynomial transformed from ϕ_i for $i = 1, \ldots, n$.

When involving no parameter, the system above can be effectively solved by means of Gröbner bases [4] or triangular decomposition [13, 22]. For parametric systems, comprehensive Gröbner bases are introduced and developed to handle the problem that parametric Gröbner bases may not remain stable under specialization [17, 21]. In addition, triangular decomposition methods can also be modified to solve parametric systems [5, 7, 19].

These methods for parametric equations can well establish conditions on the parameters for the classification of solutions. However, they are not very efficient without consideration of the special structure of equations over $\mathbb{F}_2$. An alternative way is to specify the parameters first and then solve the parameter-free system directly. This method is feasible because there are only finitely many points in the parameter space and it may be efficient if the number of parameters is relatively small.

As an example of the latter method, we consider a Boolean network studied in [1, 11], which has 14 variables and 7 parameters. For solving the corrsponding parametric Boolean equation system, we traverse all the possible 2^7 specifications of the parameters and compute the Gröbner basis for each specification. From these Gröbner bases, all the solutions can be easily obtained and thus we can completely determine the classification of the steady states. The following table illustrates the number of specifications corresponding to different numbers of steady states.

Number of Steady States	0	1	2	4
Number of Specifications	37	31	24	36

For example, for the specification $[1, 1, 0, 1, 0, 1, 0]$, the Gröbner basis $\langle 1 \rangle$ indicates the system has no steady state, while for $[1, 1, 1, 1, 1, 1, 1]$ the system has four steady states.

This experiment was made in PolyBoRi 2.6.2 [3] running on Pentium(R) P8700 CPU 2.53GHz with 1.89G RAM under Ubuntu 9.04 OS. All the traversal computation of 2^7 cases took 1.28 seconds. In spite of its efficiency, this traversal method cannot provide a good representation of the solution classification but a list of all the 2^7 specifications.

4 Future Work

Our future work may include the analysis of stability for finite dynamical systems and automated classification of steady states. Furthermore, improvement of the techniques

for analyzing high-dimensional discrete biological systems efficiently, the modification and specialization of symbolic parametric methods, and the detection of good representation of results from traversal computation for Boolean networks are also some of the problems that remain for future research.

References and Notes

[1] R. Albert and H. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *J. Theor. Biol.*, 223:1–18, 2003.

[2] L. J. S. Allen. Some discrete-time SI, SIR, and SIS epidemic models. *Math. Biosci.*, 124:83–105, 1994.

[3] M. Brickenstein and A. Dreyer. PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials. *J. Symb. Comput.*, 44(9):1326–1345, 2009.

[4] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, Universität Innsbruck, Austria, 1965.

[5] C. Chen, O. Golubitsky, F. Lemaire, M. Moreno Maza, and W. Pan. Comprehensive triangular decomposition. In *CASC 2007*, pages 73–101. Springer-Verlag, Berlin Heidelberg, 2007.

[6] O. Galor. *Discrete Dynamical Systems*. Springer, Berlin, 2007.

[7] X. S. Gao and S. C. Chou. Solving parametric algebraic systems. In *ISSAC 1992*, pages 335–341. ACM Press, New York, 1992.

[8] O. L. R. Jacobs. *Introduction to Control Theory*. Clarendon Press, Oxford, 1974.

[9] E. I. Jury. *Inners and Stability of Dynamic Systems*. John Wiley, New York, 1974.

[10] S. Kauffman. *The Origin of Order: Self-Organization and Selection in Evolution*. Oxford Univ. Press, New York, 1993.

[11] R. Laubenbacher and B. Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *J. Theor. Biol.*, 229:523–537, 2004.

[12] R. Laubenbacher and B. Sturmfels. Computer algebra in systems biology. *Am. Math. Mon.*, in press, 2009.

[13] D. Lazard. A new method for solving algebraic systems of positive dimension. *Discrete Appl. Math.*, 33(1-3):147–160, 1991.

[14] W. Niu and D. Wang. Algebraic analysis of bifurcation and limit cycles for biological systems. In *AB 2008*, pages 156–171. Springer-Verlag, Berlin Heidelberg, 2008.

[15] W. Niu and D. Wang. Algebraic approaches to stability analysis of biological systems. *Math. Comput. Sci.*, 1(3):507–539, 2008.

[16] R. C. Oldenbourg and H. Sartorius. *The Dynamics of Automatic Control*. The American Society of Mechanical Engineers, New York, 1948.

[17] A. Suzuki and Y. Sato. A simple algorithm to compute comprehensive Gröbner bases using Gröbner bases. In *ISSAC 2006*, pages 326–331. ACM Press, New York, 2006.

[18] T. Tamura and T. Akutsu. Algorithms for singleton attractor detection in planar and non-planar AND/OR Boolean networks. *Math. Comput. Sci.*, 2(3):401–420, 2009.

[19] D. Wang. Computing triangular systems and regular systems. *J. Symb. Comput.*, 30(2):221–236, 2000.

[20] D. Wang and B. Xia. Stability analysis of biological systems with real solution classification. In *ISSAC 2005*, pages 354–361. ACM Press, New York, 2005.

[21] V. Weispfenning. Comprehensive Gröbner bases. *J. Symb. Comput.*, 14(1):1–29, 1992.

[22] W.-T Wu. *Mathematics Mechanization*. Science Press/Kluwer, Beijing, 2000.

Optimization and Synthesis for a Mechatronic Network

Fu-Cheng Wang[1*], Hsiang-An Chan[1], Jason Zheng Jiang[2], and Malcolm C. Smith[2]

[1] Department of Mechanical Engineering, National Taiwan University, No.1, Sec. 4, Roosevelt Road, Taipei 10617, Taiwan. fcw@ntu.edu.tw, cardiel740@yahoo.com

[2] Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, U.K. zj219@cam.ac.uk, mcs@cam.ac.uk

Abstract

This paper applies a novel mechatronic system to vehicle suspensions. The proposed mechatronic system consists of a ball-screw inerter and permanent magnet electric machinery (PMEM), such that the system impedance is a combination of mechanical and electrical networks. Then we apply linear matrix inequalities (LMI) to optimize system performance, and discuss network synthesis of the obtained optimal impedances. The results demonstrate the effectiveness of the mechatronic system and newly introduced network synthesis methods

1 Introduction

A new mechanical element, called an inerter, was proposed in [1] to substitute for the mass element in the mechanical/electrical analogy, to improve system performance. The inerter has been applied to car suspensions [2], motorcycle steering [3], train suspensions [4, 5] and building vibration control [6], and showed great performance improvements. Moreover, LMI methods were applied in [7] to further increase the performance benefits, by allowing higher-order positive real impedances. However, the synthesis of a high-order mechanical network is difficult due to the limitation on the volume and weight of systems. Therefore, a novel mechatronic system was proposed in [8], which consists of a ball-screw inerter and PMEM, such that the system impedance can be realized through a combination of mechanical structures and electrical circuits. Consequently, a high-order impedance function can be synthesized by a basic mechanical layout and high-order electrical circuits. Applying the mechatronic networks and LMI approaches to vehicle suspensions, significant performance improvement can be achieved. In addition, the obtained optimal impedances can be realized using the traditional [9] or newly-developed synthesis methods [10, 11]. This paper is arranged as follows: Section 2 introduces the mechatronic system and represents it as a network. Section 3 applies the mechatronic network to vehicle suspensions for performance optimization using LMI approaches. Section 4 applies three synthesis methods to realize the obtained optimal impedances. Finally, we draw some conclusions in Section 5.

2 The Mechatronic Network System

The mechatronic system was proposed in [8], which consists of a ball-screw inerter and a coaxial PMEM. Therefore, suitable electrical circuits can be connected to the PMEM, such

*Correspondence to: National Taiwan University, Taiwan, tel: +886 2 33662680, fax: +886 2 23631755.

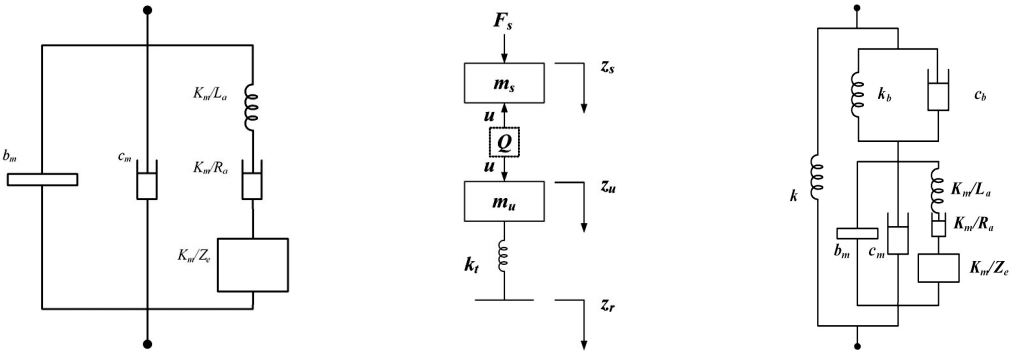
that the overall system impedance Y_{ms} is a combination of the mechanical and electrical impedances, as follows:

$$Y_{ms} = \frac{\hat{F}(s)}{\hat{v}(s)} = b_m s + c_m + \frac{K_m}{R_a + sL_a + Z_e(s)} \tag{1}$$

where b_m, c_m, and K_m are the inertance, damping rate, and admittance gain of the mechatronic strut, respectively, as in the following:

$$b_m = (2\pi/P)^2(J_m + J),\ c_m = (2\pi/P)^2 B_m,\ K_m = (2\pi/P)^2 k_t k_e,$$

in which P is the pitch of the ball-screw, J is the mass moment of inertia of the ball-screw inerter, J_m is the mass moment of inertia of the PMEM, B_m is the damping coefficient of the PMEM, k_e is the inductive voltage constant of the PMEM, and k_t is the inductive torque constant of the PMEM. Note that Y_{ms} can be divided into two parts. The first part, $b_m s + c_m$, can be considered the mechanical inerter and damper of the mechatronic system. The second part, $K_m/(R_a + sL_a + Z_e(s))$, can be regarded as the electronic admittance of the system. Using mechanical and electrical analogies, (1) can be represented by an equivalent mechanical network, as shown in Figure 1(a).



(a) Network.

(b) A quarter-car.

(c) The LMIS3 strut.

Figure 1: The application of mechatronic struts to vehicle suspensions [8].

3 Performance Optimization of a Quarter-Car Model

A quarter-car model is illustrated in Figure 1(b), with the suspension force $\hat{u}(s) = Q(s)s(\hat{z}_s - \hat{z}_u)$, which depends on the applied suspension layouts with admittance $Q(s)$. To evaluate the system performance, we optimized two performance measures J_1 and J_3 (defined in [12]) using six suspension layouts, and further applied LMI approaches (see Theorem 1 of [5]) for numerical optimization of these layouts.

Setting the system parameters $m_s = 250$kg, $m_u = 35$kg, $k_t = 150$kN/m, $V = 25$m/s, $\kappa = 5 \times 10^{-7}\text{m}^3\text{cycle}^{-1}$, $R_a = 2.3\Omega$, $L_a = 0.7$mH, $K_m = 7056$ VNs/A/m, we found that

the LMIS3 layout, as illustrated in Figure 1(c), provides the most performance benefits. In addition, the mechatronic system can significantly improve the performance measures for both soft and stiff systems, compared to the mechanical inerter layouts, which were useful only for the stiff systems [2, 4, 5, 12]. Furthermore, setting stiffness $k = 50$kN/m, the LMIS3 layout with the following electrical impedances [8]:

$$Z_{e,J_1}^{2nd} = \frac{1.665 \times 10^5 s^2 + 5.776 \times 10^5 s + 5.466 \times 10^7}{s^2 + 1.544 \times 10^6 s + 0.342}, \quad (2)$$

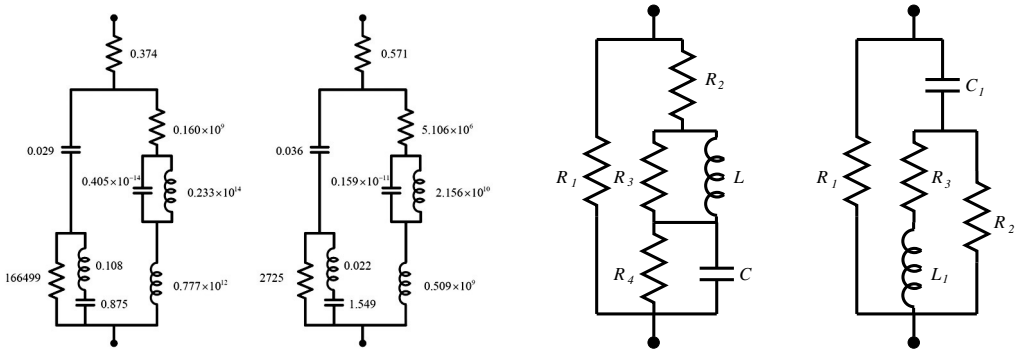
$$Z_{e,J_3}^{2nd} = \frac{2.726 \times 10^3 s^2 + 7.060 \times 10^4 s + 3.454 \times 10^6}{s^2 + 1.235 \times 10^5 s + 0.676}, \quad (3)$$

was shown to be optimal for the performance measures.

4 Network Synthesis for Mechatronic Systems

The obtained electrical impedances of (2-3) can be realized by the Brune or Bott-Duffin methods [9]. However, these two methods are not ideal, in that Brune realization needs perfect transformers, and Bott-Duffin realization uses a large number of elements. Therefore, Chen and Smith [10] proposed five efficient networks to synthesize certain second-order passive networks with one capacitor, one inductor, and four resistors. The results were improved in [11] with the recently introduced concept of regularity, such that only one capacitor, one inductor, and three resistors are necessary.

To avoid the use of perfect transformers with the Brune realization, Bott and Duffin [9] proposed to use nine elements to synthesize a second-order transfer function. The realization procedures were summarized in (Figure A1 of [5]), such that the obtained impedances (2-3) can be realized as in Figure 2(a). Note that these realizations not only use nine elements for a second-order transfer function (with only five parameters), but also use unreasonable component values. Therefore, it is necessary to develop better network synthesis for the obtained system impedances.



(a) Bott-Duffin. (b) Chen-Smith. (c) Jiang-Smith.

Figure 2: Network realizations.

To overcome the shortages of Bott-Duffin realization, Chen and Smith [10] proposed a new network synthesis by which certain second-order passive networks can be realized with one capacitor, one inductor, and four resistors (corresponding to one inerter, one spring and four dampers in mechanical systems). Following the proposed procedures, the optimal impedances (2-3) can be realized as in Figure 2(b), with the following parameters: (1) Z_{e,J_1}^{2nd}: with $R_1 = 2.303 \times 10^9\Omega$, $R_2 = 0.374\Omega$, $R_3 = 1.666 \times 10^5\Omega$, $R_4 = 1.711 \times 10^8\Omega$, $C = 0.0284$F, $L = 0.1079$H; (2) Z_{e,J_3}^{2nd}: with $R_1 = 1.318 \times 10^8\Omega$, $R_2 = 0.571\Omega$, $R_3 = 2.725 \times 10^3\Omega$, $R_4 = 5.312 \times 10^6\Omega$, $C = 0.0358$F, $L = 0.0221$H. We note that this new realization is better than Bott-Duffin in terms of using fewer elements with more reasonable values. Nevertheless, it uses six elements for a second-order transfer function which has only five parameters. Therefore, an interesting question to ask is whether an even simpler realization exists? This question was answered in the following.

Jiang and Smith [11] proposed a simpler network synthesis method which uses only five components for the second-order transfer functions which are regular (see Lemma 5 of [11]). Applying the procedures, the impedances (2-3) can be realized as in Figure 2(c) with the following parameters: (1) Z_{e,J_1}^{2nd}: with $R_1 = 1.598 \times 10^8\Omega$, $R_2 = 1.667 \times 10^5\Omega$, $R_3 = 0.374\Omega$, $C = 0.0282$F, $L = 0.1078$H; (2) Z_{e,J_3}^{2nd}: with $R_1 = 5.109 \times 10^6\Omega$, $R_2 = 2.727 \times 10^5\Omega$, $R_3 = 0.572\Omega$, $C = 0.0358$F, $L = 0.0221$H. It is interesting that the required network elements are fewer than the Chen-Smith realization, and with similar component values.

5 Concluding Remarks

This paper has applied a novel mechatronic system to vehicle suspension control to illustrate its performance benefits. In addition, the obtained optimal impedances were realized by the Bott-Duffin and two newly-developed methods. From the results, the proposed mechatronic system was deemed effective and the applied network synthesis was shown to be efficient.

References and Notes

[1] M.C. Smith, Synthesis of mechanical networks: The Inerter, IEEE Transactions on Automatic Control, vol. 47, no. 10, pp. 1648-1662, 2002.

[2] M.C. Smith and F.C. Wang, Performance benefits in passive vehicle suspensions employing inerters, Vehicle System Dynamics, vol. 42, no. 4, pp. 235-257, 2004.

[3] S. Evangelou, D.J.N. Limebeer, R.S. Sharp and M.C. Smith, Steering compensation for high-performance motorcycles, Proc. IEEE Conference on Decision and Control, pp. 749-754, 2004.

[4] F.C. Wang, C.H. Yu, M.L. Chang and M.S. Hsu, The performance improvements of train suspension systems with inerters, Proc. IEEE Conference on Decision and Control, pp. 1472-1477, 2006.

[5] F.C. Wang, M.K. Liao, B.H. Liao, W.J. Su and H.A. Chan, The performance improvements of train suspension systems with mechanical networks employing inerters, Vehicle System Dynamics, Vol.47, no.7, pp.805-830, 2009.

[6] F.C. Wang and C.W. Chen, Performance analyses of building suspension control with inerters, Proc. IEEE Conference on Decision and Control, pp. 3786-3791, 2007.

[7] C. Papageorgiou and M.C. Smith, Positive real synthesis using matrix inequalities for mechanical networks: Application to vehicle suspension, IEEE Trans. on Control Systems Technology, vol. 14, no. 3, pp. 423-435, 2006.

[8] F.C. Wang and H.A. Chan, Mechatronic suspension design and its applications to vehicle suspension control, Proc. IEEE Conference on Decision and Control, pp. 3769-3774, 2008.

[9] J.E. Storer, Passive Network Synthesis. New York: McGraw-Hill, 1957.

[10] M.Z.Q. Chen and M.C. Smith, Electrical and Mechanical Passive Network Synthesis, Proc. Workshop on Recent Advances on Control and Learning, pp. 35-50, 2008.

[11] J.Z. Jiang and M.C. Smith, Regular Positive-Real Functions and Passive Networks Comprising Two Reactive Elements, Proc. European Control Conference, pp. 219-224, Budapest, Hungary, August 23-26, 2009.

[12] F.C. Wang and W.J. Sue, The impact of inerter nonlinearities on vehicle suspension control, Vehicle System Dynamics, vol. 46, no. 7, pp. 575-595, 2008.

Algebraic approaches to underdetermined systems

Hiroshi Yoshida[1*], and Kinji Kimura[2]

[1] Faculty of Mathematics, Kyushu University,
Itou, Motouoka 744, Nishi-ku FUKUOKA 819-0395 Japan.
phiroshi@math.kyushu-u.ac.jp

[2] Graduate School of Informatics, Kyoto University,
Yoshida-Honmachi, Sakyo-ku, KYOTO 606-8501 Japan.
kkimur@amp.i.kyoto-u.ac.jp

Abstract

It is difficult, but important, to analyze nonlinear chemical reactions away from equilibrium. For accurate analysis of such chemical systems, we need to deal directly with the nonlinear terms. This becomes more difficult when rate constants cannot be determined in a single experiment; in other words, the system is underdetermined. To overcome this underdetermination, we propose a method to combine the results of multiple experiments. Nevertheless, multiple experiments usually pose another difficulty in that the solution to the system of equations has multiple solutions. These two difficulties lead to confusion as to whether the rate constants can be intrinsically determined by experiment or not. Here, to analyze such a system, we use prime ideal decomposition, and discuss a scheme for generalized multiple experiments as well as an efficient calculation using resultants.

1 Introduction

We often analyze nonlinear chemical reactions such as the Michaelis–Menten or Hill type enzymatic reactions. These reactions are described as a system of differential equations including nonlinear terms that make the analysis difficult. To overcome this difficulty, conventional numerical methods or analytical methods under various simplifying assumptions such as linearization or system equilibrium are often used. Another difficulty is that the system is sometimes *underdetermined*. A consequence of underdetermination is that insufficient data from a single experiment exist to determine concrete values for rate constants. We proposed a method to overcome such underdetermination by combining two experiments in a previous paper [3]. In this paper, we also propose an approach for determining the rate constants by combining multiple experiments, each of which is an underdetermined system.

The combination of multiple experiments, however, yields another difficulty in which the solution to the system of equations can be decomposed into multiple distinct solutions. For instance, imagine that a system of equations describing some experiment yields the algebraic equations: $\{z^2-2, y^2+2y-1, xy+xz-yz+x-z-2\}$. The solution to these equations can be decomposed into the solutions of the following two systems: $\{z^2-2, y+z+1\}$ and $\{z^2-2, y-z+1, x-z\}$. We cannot decide the variables, x, y and z, with the former, but we can decide with the latter. Further, under a physiologically acceptable condition, $x>0, y>0$, & $z>0$, this decomposition means the variables are identifiable ($x = z = \sqrt{2}, y = \sqrt{2}-1$). Therefore, it is necessary to perform decomposition to analyze the system because we cannot know beforehand whether the algebraic equations have a unique solution. Such decomposition of algebraic equations is called *prime ideal decomposition*.

*Correspondence to: H. Yoshida, (Kyushu Univ., Itou)

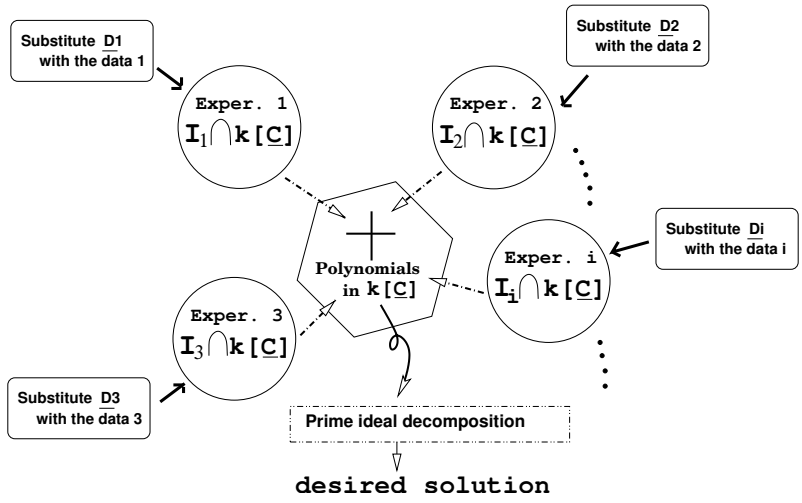


Figure 1: The schematic illustration of procedure w.r.t. multiple experiments.

2 Theorem and method for many parameters and variables

To overcome underdetermination of a system, we combine multiple experiments. But, multiple experiments yields many parameters and variables. Here we present a theorem and a procedure to avoid increasing the number of variables, even when we need to combine multiple experiments. From the viewpoint of *ideal* theory, the procedure w.r.t. two experiments A and B can be described as the following theorem:

$$V\left((I_A + I_B) \cap \Bbbk[\underline{C}]\right) = V\left(\left(I_A \cap \Bbbk[\underline{C}]\right) + \left(I_B \cap \Bbbk[\underline{C}]\right)\right), \tag{1}$$

where $V(J)$ denotes the affine variety with respect to the ideal J. In this theorem, I_A and I_B denote the ideals associated with Experiments A and B, respectively; and $\Bbbk[\underline{X}]$ designates the ring obtained by adjoining the variables $\underline{X}$ to field $\Bbbk$. Further, $\underline{C}$ denotes the set of the maximum common variables between the experiments A and B, while $\underline{A}$ and $\underline{B}$ denote the distinct sets of intrinsic variables of the experiments A and B, respectively.

By recursion, Theorem (1) can be extended to the theorem below:

$$V\Big(\big(\sum_{i=1}^{n} I_i\big) \cap \Bbbk[\underline{C}]\Big) = V\Big(\sum_{i=1}^{n} (I_i \cap \Bbbk[\underline{C}])\Big), \tag{2}$$

where I_i denotes the ideal associated with Experiment i; and $\underline{C}$ the set of the maximum common variables of all of the experiments $i (1 \leq i \leq n)$. The generalized procedure w.r.t. multiple experiments using Theorem (2) is illustrated in Fig. 1. Let I_i denote the ideal generated by polynomials in $k(\underline{D_i})[\underline{A_i}, \underline{C}]$, where $\underline{D_i}$ denotes the parameters determined with the corresponding experimental data; and $\underline{A_i}$ denotes the sets of intrinsic variables of Experiment i. The procedure is summarized as follows:

(i) For each experiment i, we substitute $\underline{D_i}$ with rational approximations of the observed data.

(ii) For each experiment i, we calculate the Gröbner basis G_i in terms of the elimination order $\underline{A_i} \gg \underline{C}$. Let Gc_i be $G_i \cap \Bbbk[\underline{C}]$, a Gröbner basis corresponding to the elimination ideal $\overline{I_i} \cap \Bbbk[\underline{C}]$.

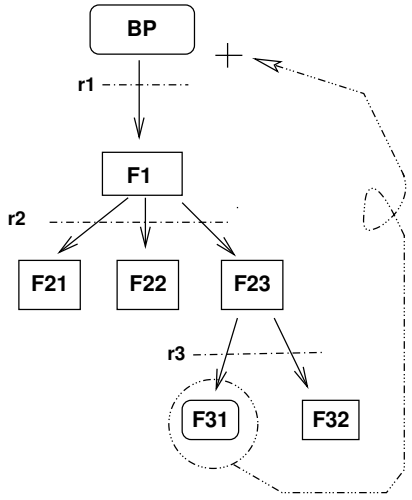


Figure 2: Resultant-factorization technique for efficient calculation.

(iii) We perform prime ideal decomposition: $\langle \sum_i \mathrm{Gc}_i \rangle = \bigcap_i P_i$.

(iv) For each prime ideal P_i obtained above, we calculate the dimension of P_i.

(v) If we find a zero-dimensional (prime) ideal, we can determine the rate constants in $\underline{C}$; otherwise, we design and add another experiment and start from step (i).

From the viewpoint of biological experiments, this procedure corresponds to a discovery of the sets of experiments by which we can determine the rate constants in $\underline{C}$ from the observed data. Therefore, once we discover such an appropriate set of experiments, we can determine the rate constants for different observed data using the same set of experiments. Further, it is assured from Theorem (2) that the rate constants thus obtained are identical to those in the original variety: $V\left((\sum_i I_i) \cap \mathbb{k}[\underline{C}]\right)$.

2.1 Resultant-factorization technique

It is cumbersome to perform prime ideal decomposition each time we obtain observed data corresponding to $\underline{D_i}$ in Fig. 1. At the present time, however, it is impossible to perform straightforward prime ideal decomposition without substituting concrete values for the parameters, one cannot perform prime ideal decomposition because of the large number of parameters and variables. That is why we substituted the observed data before prime ideal decomposition. In addition, we think that the larger the number of multiple experiments, the more difficult straightforward decomposition becomes [2].

Therefore, based on [1], we developed a technique to arrive at the desired zero-dimensional affine variety as illustrated in Fig. 2. Let $BP = \{BP_i \ \ 1 \leq i \leq n\}$ be the original set of polynomials. Let F_1 be the set of $n-1$ resultants of polynomials BP_j $(1 \leq j \leq n, i \neq j)$, for some BP_i $(1 \leq i \leq n)$ in a variable, say, r_1. It is reasonable to select a variable r_1 such that BP_i has the smallest degree in r_1. If, for instance, some element f in F_1 can be factorized into mutually disjoint polynomials f_1, f_2, and f_3, $\langle F_1 \rangle$ can be decomposed into $\langle F_1, f_1 \rangle \cap \langle F_1, f_2 \rangle \cap \langle F_1, f_3 \rangle$. As a result of the factorization, the next resultant set for $\langle F_1, f_1 \rangle$ in r_2, denoted by F_{21}, can usually be described by a smaller set of

polynomials. Further, when one obtains a factorized term like $(y_1 + y_2^2)(z - y_3^2 + 3)^3$ with the rate constants y_1, y_2, it is sufficient to take account only of $(z - y_3^2 + 3)$, because the rate constants are physiologically assured to be positive ($y_1 > 0 \wedge y_2 > 0 \Rightarrow y_1 + y_2^2 > 0$) and the radical suffices. We can also ignore a factor like $(y_1 - y_2)$ when we adopt a presupposition $y_1 \neq y_2$ from biological or physiological knowledge. These simplifications allow us to prune the branches during the resultant-factorization process.

When we reach an appropriate set (F_{31}, in the figure), we can efficiently determine all of the rate constants by using the ideal $\langle BP \rangle + \langle F_{31} \rangle$ as illustrated in Fig. 2. This technique is given in http://sites.google.com/site/codes86/.

In the presentation of MACIS-2009, we are going to show an example in which the proposed theorem and technique methods here are used and to show the range of applicability of them.

Acknowledgements

We express our gratitude to Prof. Kazuhiro Yokoyama and Prof. Shunsuke Takagi for valuable suggestions on Theorem (1). We also thank Prof. Miwa Yoshihiro (Univ. of Tsukuba) for valuable discussions of experiments. This study was supported in part by the Program for Improvement of Research Environment for Young Researchers from Special Coordination Funds for Promoting Science and Technology (SCF) commissioned by the Japan Science and Technology Agency (JST).

References and Notes

[1] Y. Kawano, K. Kimura, H. Sekigawa, M. Noro, K. Shirayanagi, M. Kitagawa, and M. Ozawa. Existence of the exact CNOT on a quantum computer with the exchange interaction. *Quantum Inf. Process.*, 4(2):65–85, 2005.

[2] Takeshi Shimoyama and Kazuhiro Yokoyama. Localization and primary decomposition of polynomial ideals. *J. Symbolic Comput.*, 22(3):247–277, 1996.

[3] Hiroshi Yoshida, Koji Nakagawa, Hirokazu Anai, and Katsuhisa Horimoto. Exact parameter determination for Parkinson's disease diagnosis with PET using an algebraic approach. In H. Anai, K. Horimoto, and T. Katsia, editors, *Algebraic Biology*, volume 4545 of *Lecture Notes in Computer Science*, pages 110–124, Heidelberg, 2007. Springer-Verlag.

Automatically Generating High-Performance Parallel Code for Atmospheric Simulation Models: Challenges and Solutions for Auto-Programming Tools

Robert van Engelen*

Department of Computer Science, Florida State University
Tallahassee, FL32306-4530, USA
engelen@cs.fsu.edu

Abstract

Traditional software design and implementation of large computational models is a difficult, tedious, and often error-prone task. Software engineers have to thoroughly analyze high-level abstract specifications of solution methods, algorithms, and data structures and carefully lower them into efficient software implementations. This software development process is iterative and repetitive as models and code are continuously refined and errors are corrected. In the Ctadel project we explored new methods for autocode generation for atmospheric models, where the designer describes the computational model in an abstract high-level specification language which is translated into highly optimized sequential and parallel code. In this extended abstract we address challenges and present solutions for auto-programming tools. We show how correctness guarantees can be achieved by autocode generation using a model-driven framework. Correctness is the primary concern. The systematic lowering of high-level specifications to executable code proceeds through the application of correctness-preserving transformations performed within an algebraic framework of application-dependent and independent program transformations. Efficiency of the target code is a secondary, but important, concern. Aggressive code optimization at the highest level is achieved by algebraic program manipulation and restructuring-compiler optimizations combined with a target machine architecture model that determines an optimal code translation strategy.

1 Introduction

Scientific software can be subject to significant change over its lifetime. Computational scientists continuously improve the accuracy and performance of numerical solution methods, algorithms, and data structures. Thus, large-scale projects that incorporate many computational methods are frequently updated with the latest state-of-the-art in the computational domain. Examples of large-scale projects include climate models, coupled ocean-atmosphere models, and weather forecast systems. A typical weather forecast for the next day, for example, requires at least a trillion (10^{12}) arithmetic operations performed by various computational modules that model the limited-area atmosphere, including solving the partial differential equations (PDEs) that govern the atmosphere, computing cloud layers, calculating solar radiation, and modeling of other dynamical processes.

Time-constrained scientific models, such as weather forecast systems that run hourly-updated weather forecasts, must make a trade-off between the accuracy of the numerical solution and the maximum amount of computing time that can be alloted to produce results. Fortunately, the rise of newer generations of ever more powerful parallel machines

*Correspondence to: Department of Computer Science, Florida State University, James J. Love bldg., Tallahassee, FL32306-4530, USA, +1(850)6440309.

has provided opportunities to redevelop the scientific software for parallel execution. By going parallel, ideally higher numerical accuracy can be achieved in the same compute time. However, it is difficult and time consuming to redevelop scientific software for high-performance machines or to port existing scientific software to these machines [2]. This is mainly due to the shortage and weakness of available development tools. Therefore, incorporating computational improvements can have a major implementation consequence. Manual modification of a model's code is a tedious and error-prone task.

In the CTADEL project [3, 4, 6] we explored new methods for autocode generation for atmospheric models, where the designer describes the computational model at an abstract high-level specification language ATMOL [6], which is translated into highly optimized sequential and parallel code [4]. We tested our approach on a large-scale weather forecast system [1]. Code generation from a domain-specific abstract problem specification has great advantages for correctness, efficiency, and portability.

Correctness Guarantees We defined a set of common rewrite rules to translate PDE-based models into low-level Fortran code optimized by applying rewrite rules on a Fortran-like symbolic intermediate code form. The rewrite rules defined at all levels only make use of the local structural information of each symbolic term that is rewritten, thereby ensuring that the correctness of each rule can be easily determined (most are simply directed forms of algebraic identities). As each rule is verifiably correct, the application of over hundred thousand rewrites on the atmospheric model yields a code that is correct, assuming that the model's specification is correct. In fact, when comparing our auto-coded implementation to the manually-developed model implementation used in the production weather forecast system, we found several programming errors in the manually developed production-quality system. One error was a programming mistake related to updating the values at the boundaries of certain array variables. Another error was found in the numerical scheme: the forecast model can deal with spherical grids only, while the model was originally intended to handle more general curvi-linear grids. A third problem was found in the model description involving the wrong units of dimensionality. By coincidence, these mistakes did not affect the quality of the weather forecast (a case of luck rather than wisdom). This illustrates the importance and usefulness of correctness guarantees in autocode generation approaches.

Efficiency Guarantees When the first compilers for Fortran were developed, no one believed that compilers could produce more efficient code than assembly programmers could. By contrast, the machine code running on today's computers is so complex to optimize that expert assembly programmers have a hard time to beat compiler-generated code. Likewise, it is conceivable that autocode generators will produce code that outperforms hand-written code. For example, parallel machines are complex to program and autocode generators can be used to generate different codes for each parallel programming paradigm and parallelization strategy. One can simply pick the best performing autocode. In fact, we observed that our auto-generated codes outperformed the original hand-written codes on several high-performance machines [4]. We used a crude model that models the target hardware architecture characteristics to derive more efficient code.

Portability Guarantees Autocode generators can adopt a different back-end generator to simplify porting. Code can be generated for new programming languages and platforms.

```
% Declare spatial and time dimensions:
space (x(i),y(j),z(k)) time t.
% Declare grid size variables n, m, and l:
n :: integer(1..infinity); m :: integer(1..infinity); l :: integer(2..infinity).
% For convenience, define macros for two grid domains spanning (i,j,k):
atmosphere := i=1..n by j=1..m by k=1..l; surface := i=1..n by j=1..m.
% Set coordinate system for symbolic derivation with chain-rule:
coordinates := [x, y]; coefficients := [h_x, h_y].
% Declare the model fields:
u     :: float dim "m/s" field (x(half),y(grid),z(grid)) on atmosphere.
v     :: float dim "m/s" field (x(grid),y(half),z(grid)) on atmosphere.
u_aux :: float dim "Pa m/s" field (x(half),y(grid),z(half)) on atmosphere.
v_aux :: float dim "Pa m/s" field (x(grid),y(half),z(half)) on atmosphere.
p     :: float(0..107000) dim "Pa" field(x(grid),y(grid),z(grid)) monotonic k(+) on atmosphere.
p_s_t :: float dim "Pa/s" field (x(grid),y(grid)) on surface.
% Define the horizontal wind velocity vector components:
V := [u_aux, v_aux].
% Equations:
p_s_t = -int(nabla .* V, z=1..l).
    V = [u, v] * d p/d z.
```

Figure 1: Specification of the surface pressure tendency

2 Model-Driven Autocode Generation with CTADEL

This section briefly describes the main concepts of the CTADEL model-driven autocode generation. More details can be found in [4, 6].

2.1 Domain Specific Modeling

Consider the *surface pressure tendency equation* and the *equation for the auxiliary horizontal wind velocity vector field* [1]:

$$\frac{\partial p_s}{\partial t} = -\int_0^1 \nabla \cdot \mathbf{V}\, dz \quad ; \qquad \mathbf{V} = \begin{pmatrix} u \\ v \end{pmatrix} \frac{\partial p}{\partial z}$$

The specification in ATMOL is shown in Fig. 1. The full specification of the weather forecast model *dynamics* is available from `www.cs.fsu.edu/~engelen/ctadel/dyn/report.html`.

2.2 User-Definable Type Systems

We implemented a generic type inference algorithm with a modular type system that supports overloading and subtyping. Three user-definable type systems are used to automatically infer Arakawa-style computational grid information (the grid type system), dimension and units (the unit type system), and value types (the basic type system). Inference on type-overloaded operators leads to the selection of an appropriate operator such as a finite difference operator on a grid half point for a centered difference. The type inference algorithm uses a *forward-backward scheme* and *iterative deepening* to compute the least number of conversion operations for a type-mismatched operator, when possible.

2.3 From Specification to Executable Parallel Code

We used a classification mechanism to define a hierarchy of properties of objects and operators. Fig. 2 depicts the hierarchy of operator classes. The idea is that rewrite rules can be applied to operators based on operator classification, rather than hardcoding each rule for each operator.

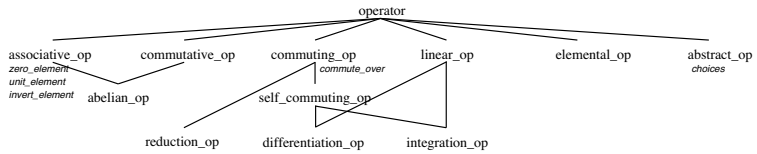


Figure 2: Operator property hierarchy

The highest level translations are performed by a rewrite rule system. A template-based code generation approach is used to lower abstract high-level operations into code for operations that are not handled easily by rewrite rules, such as lowering of reductions, scans, and search functions. The type system plays an important role in template instantiation, as templates are picked based on type. For example, a reduction over an associative operator can be parallelized using a template instantiation of a parallel reduction loop:

```
reduce(Expr :: _T, Iter :: domain(index), Op :: associative(_T->_T->_T)) :: _T function
{ reduce := unit_element(Op);
  (reduce := apply_op(Op, reduce, Expr)) forall Iter
}.
```

Instantiating this template leads to low-level assignments and a forall-loop to reduce the value of the expression over a domain, e.g. for summing a quadrature. Various strategies to automatically parallelize reductions and scans (prefix sums) were explored [5].

3 Conclusions

Auto-programming tools are increasingly more important to help develop certified software and mission-critical systems. We used a model-driven approach that utilizes verifiably correct translations of an atmospheric model into an efficient parallel code.

References and Notes

[1] E. Kållén (ed.). *HIRLAM Documentation Manual System 2.5.* Norrköping, Sweden, June 1996. Available from SMHI, S–60176.

[2] C. Pancake and D. Bergmark. Do parallel languages respond to the needs of scientific programmers? *IEEE Computing*, 23(12):13–23, December 1990.

[3] P. van der Mark, R. van Engelen, K. Gallivan, and W. Dewar. A case study for automatic code generation on a coupled ocean-atmosphere model. In *2002 International Conference on Computational Science (ICCS)*, pages 419–428, Amsterdam, The Netherlands, April 2002. Springer Verlag. Lecture Notes in Computer Science No. 2329.

[4] R.A. van Engelen, L. Wolters, and G. Cats. CTADEL: A generator of multi-platform high performance codes for PDE-based scientific applications. In 10^{th} *ACM International Conference on Supercomputing (ICS)*, pages 86–93, New York, 1996. ACM Press.

[5] R.A. van Engelen, L. Wolters, and G. Cats. PDE-oriented language compilation and optimization with CTADEL for parallel computing. In *HIPS'97* 2^{nd} *International Workshop*, pages 105–109, Geneva, Switzerland, April 1997. IEEE Computer Society Press.

[6] Robert van Engelen. ATMOL: A domain-specific language for atmospheric modeling. *Journal of Computing and Information Technology*, 9(4):289–304, December 2001.

SPIRAL and Beyond: Automatic Derivation and Optimization of DSP Algorithms and More

Jeremy Johnson[1]

[1] Department of Computer Science
Drexel University
Philadelphia, PA 19104 USA
jjohnson@cs.drexel.edu

Abstract

The SPIRAL system (www.spiral.net) is a tool for automatically deriving, implementing and optimizing digital signal processing (DSP) algorithms, in particular fast transform algorithms such as the fast Fourier transform (FFT). SPIRAL is capable of generating optimized implementations on a variety of platforms including SSE, multicore, Cell, GPU, distributed memory parallel processors, and FPGA, and has produced some of the fastest implementations of these algorithms on these platforms (SPIRAL is used by Intel in the implementation of their MKL and IPP libraries). SPIRAL uses a domain specific language, based on an algebraic formulation of DSP algorithms, and rewrite rules to generate a large number of implementations and uses intelligent search to find fast implementations. This talk provides an overview of automated generation of DSP algorithms using the SPIRAL system, briefly discusses the use of algebraic techniques in the generation of DSP algorithms, and ends with a discussion of extensions of SPIRAL that can be used for more general algorithms.

1 Introduction

Tuning high-performance libraries for important mathematical kernels such as matrix multiplication and the fast Fourier transform is a time consuming process. Algorithms need to be modified and restructured to adapt to architectural features of the platform the code will run on and the code has to be carefully tuned to fully utilize these features and maximize performance. The code tuning requires extensive experimentation and tweaking to optimize performance since it is very difficult to predict the optimal implementation on a given platform. Since the code tuning process must be reapplied for each new platform, performing it by hand requires extensive manpower, and automated techniques are strongly suggested. A general approach to automating this process is to provide a set of parameterized code generators, capable of generating a large number of alternative implementations, and tools for intelligently searching for the parameter settings that optimize performance on a given platform. The search process can be driven by empirical run times, assuming the desired platform is available, or estimates of runtime obtained through a performance model. Examples of such automated tools are ATLAS [1, 3] for dense matrix multiplication, SPARSITY [2, 3] for sparse-matrix product, and FFTW [4, 5, 6] for the fast Fourier transform.

These systems are hard wired to specific algorithms and a specific set of tuning parameters. Consequently they are difficult to extend and adapt to new platforms and new algorithms. In contrast the SPIRAL system provides a language based framework for tuning a large number of DSP algorithms on a wide variety of computing platforms.

*Department of Computer Science, Drexel University, Philadelphia, PA 19104, USA.

2 The SPIRAL system

The SPIRAL system [7, 8] begins with the observation that many fast transform algorithms can be represented as structured sparse matrix factorizations of the corresponding transform matrix. For example, let

$$\mathrm{DFT}_n = \left[\omega_n^{k\ell} \mid k, \ell = 0, \ldots, n-1\right],$$

where $\omega_n = e^{2\pi i/n}$ denotes a primitive root of unity, denote the n-point discrete Fourier transform (DFT) matrix. Let $n = rs$, then the divide and conquer step in the FFT [9], can be represented as the matrix factorization

$$\mathrm{DFT}_n = (\mathrm{DFT}_r \otimes \mathrm{I}_s) \cdot \mathrm{T}_s^n \cdot (\mathrm{I}_r \otimes \mathrm{DFT}_s) \cdot \mathrm{L}_r^n, \tag{1}$$

where $\otimes$ denotes the Kronecker or tensor product, T is a diagonal matrix called the twiddle matrix, and L is a special permutation matrix called stride permutation [10]. When $r = s = 2$,

$$\begin{aligned}
\mathrm{DFT}_4 &= (\mathrm{DFT}_2 \otimes \mathrm{I}_2) \cdot \mathrm{T}_2^4 \cdot (\mathrm{I}_2 \otimes \mathrm{DFT}_2) \cdot \mathrm{L}_2^4 \\
&= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned}$$

The Cooley-Tukey factorization 1, can be interpreted as a rewrite rule that can be applied repeatedly to a DFT of a specified size to derive an algorithm - a particular breakdown strategy - for computing the given DFT. The sequence of applications of the rule is encoded as a *ruletree* which can be translated into a formula in the domain specific language SPL [11] and compiled with a special-purpose compiler into efficient code for computing the DFT. Since many important DSP transforms can be derived by similar breakdown rules and encoded by similar matrix formulas, the SPIRAL system can be used to derive and implement a large number of important DSP transforms.

3 Evolution of the SPIRAL system

A key observation in [10] is that the mathematical constructs, such as the tensor product and stride permutations, occuring in the matrix formulas in DSP algorithms and encoded in SPL have interpretations related to different architectural features such as indexing and memory operations, vectorization and parallelism. This observation allows indexing operations to be simplified and loops merged [12], code to be vectorized with different vector sizes and automatic parallelization with perfect load balance, no false sharing, and varying granularity to be obtained mathematically through formula manipulation. Thus the same SPIRAL framework can be used to obtain optimized code in a wide variety of computing platforms such as SSE [14], multicore [13], distributed memory parallel computers [15], and FPGA [16].

SPIRAL was originally created to generate and optimize code for linear transforms of a fixed size. The restriction to fixed size transforms is fine for applications that use transforms of a specific size (e.g. JPEG, MPEG, HDTV); however, this can pose substantial problems when generating code for general libraries such as FFTW. This restriction has been recently removed through work in [17] where starting with rewrite rules such as 1 an entire library like FFTW can be automatically generated and tuned.

The restriction to linear transforms has also been recently removed, allowing algorithms with both multiple inputs and outputs and non-linear computation (e.g. matrix multiplication, convolution, Viterbi decoding, Synthetic Aperture Radar, sorting networks)[19]. The importance of the tensor product and the utility of formula manipulation techniques used by SPIRAL in multilinear computations was suggested earlier in [18] where a tensor product formulation of Strassen's matrix multiplication algorithm was presented. These ideas were used in the automatic derivation and optimization of convolution algorithms [20]. Additional papers on the use of symbolic algebra in the derivation of signal processing algorithms can be found in the special issue of the Journal of Symbolic Computation Volume 37 Issue 2.

Acknowledgements

The work was supported in part by grants from DARPA, NSF, and Intel. Material was provided by Franz Francheti, Yevgen Voronenko, Markus Püschel and the rest of the SPIRAL team (www.spiral.net).

References and Notes

[1] R. C. Whaley, A. Petitet, and J. J. Dongarra, "Automated empirical optimization of software and the ATLAS project," Parallel Computing, vol. 27, no. 12, pp. 335, 2001.

[2] E.-J. Im, K. Yelick, and R. Vuduc, "Sparsity: Optimization framework for sparse matrix kernels," International Journal of High Performance Computing Applications, vol. 18, no. 1, 2004.

[3] J. Dongarra, V. Eijkhout, E. Fuentes, J. Demmel, R. Vuduc, and K. Yelick, "Self Adapting Linear Algebra Algorithms and Software," Proceedings of the IEEE special issue on "Program Generation, Optimization, and Adaptation," Vol. 93, No. 2, 2005, pp. 293-312.

[4] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT," in ICASSP 98, vol. 3, 1998, pp. 13811384, www.fftw.org.

[5] M. Frigo, "A Fast Fourier Transform Compiler," in Proc. PLDI, 1999, pp. 169180.

[6] M. Frigo and S. G. Johnson, "The Design and Implementation of FFTW 3," Proceedings of the IEEE special issue on "Program Generation, Optimization, and Adaptation," Vol. 93, No. 2, 2005, pp. 216-231.

[7] Markus Püschel, José M. F. Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo SPIRAL: Code Generation for DSP

Transforms Proceedings of the IEEE special issue on "Program Generation, Optimization, and Adaptation," Vol. 93, No. 2, 2005, pp. 232-275.

[8] M. Püschel, B. Singer, J. Xiong, J. M. F. Moura, J. Johnson, D. Padua, M. Veloso, and R. W. Johnson, "SPIRAL: A Generator for Platform-Adapted Libraries of Signal Processing Algorithms," International Journal of High Performance Computing Applications , vol. 18, no. 1, pp. 2145, 2004.

[9] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. of Computation, 19:297–301, 1965.

[10] J. Johnson, R. W. Johnson, D. Rodriguez, and R. Tolimieri. "A Methodology for Designing, Modifying, and Implementing Fourier Transform Algorithms on Various Architectures," IEEE Trans. Circuits Sys., 9, 1990.

[11] J. Xiong, J. Johnson, R. Johnson, and D. Padua, "SPL: A Language and Compiler for DSP Algorithms," in Proc. PLDI, 2001, pp. 298308.

[12] Franz Franchetti, Yevgen Voronenko and Markus Püschel, "Formal Loop Merging for Signal Transforms," Proc. Programming Languages Design and Implementation (PLDI), pp. 315-326 , 2005.

[13] Franz Franchetti, Yevgen Voronenko and Markus Püschel, "FFT Program Generation for Shared Memory: SMP and Multicore," Proc. Supercomputing (SC), 2006.

[14] Franz Franchetti, Yevgen Voronenko and Markus Püschel, "A Rewriting System for the Vectorization of Signal Transforms," Proc. High Performance Computing for Computational Science (VECPAR), Lecture Notes in Computer Science, Springer, Vol. 4395, pp. 363-377, 2006.

[15] Andreas Bonelli, Franz Franchetti, Juergen Lorenz, Markus Püschel and Christoph W. Ueberhuber, "Automatic Performance Optimization of the Discrete Fourier Transform on Distributed Memory Computers," Proc. International Symposium on Parallel and Distributed Processing and Application (ISPA), Lecture Notes In Computer Science, Springer, Vol. 4330, pp. 818-832, 2006.

[16] Peter A. Milder, Franz Franchetti, James C. Hoe and Markus P"uschel, "Formal Datapath Representation and Manipulation for Implementing DSP Transforms," Proc. Design Automation Conference (DAC), 2008.

[17] Yevgen Voronenko, "Library Generation for Linear Transforms," PhD. thesis, Electrical and Computer Engineering, Carnegie Mellon University, 2008.

[18] R. W. Johnson, C. H. Huang and J. R. Johnson, "Multilinear algebra and parallel programming," The Journal of Supercomputing, Vol. 5, Oct. 1991.

[19] Franz Franchetti, Frédéric de Mesmay, Daniel McFarlin and Markus Püschel, "Operator Language: A Program Generation Framework for Fast Kernels", Proc. IFIP Working Conference on Domain Specific Languages (DSL WC), Lecture Notes in Computer Science, Springer, Vol. 5658, pp. 385-410, 2009.

[20] Jeremy R. Johnson, Anthony F. Breitzman, "Automatic derivation and implementation of fast convolution algorithms", Journal of Symbolic Computation, Vol. 37, No. 2, pp. 261-293, 2004.

A logic-based approach to the implementation of medical knowledge mining

Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering and Knowledge Discovery (DEKD) Research Unit
School of Computer Engineering, Suranaree University of Technology
Nakhon Ratchasima 30000 Thailand
{nittaya, kerdpras}@sut.ac.th

Abstract

Medical knowledge mining is a recently emerging area of research that provides applied scientists and practitioners with tools for processing and analysis of massive data sets in medical domains. The ultimate goal of applying data mining technology to medicine and life science is to discover effective knowledge. Induced knowledge such as cancer implication is important not only to increase accurate diagnosis and successful treatment, but also to enhance safety and reduce medication-related errors. Such implicit knowledge can be achieved through the availability of the knowledge-mining system. In this paper, we present the design and implementation of a medical knowledge-mining system. Our system implementation is based on a solid foundation of mathematical logic. The expressive power of logic-based language and the effective pattern-matching feature are essential functions for the development of knowledge-intensive tasks. We demonstrate the design framework of our proposed system, the program coding, and also the running results on the discharge-decision data of patients after operation. The high-level abstraction of logic-based language results in the concise coding. Moreover, the first-order logic formalism facilitates the automatic reuse of induced knowledge as conditional clauses in other applications.

1 Introduction

Modern healthcare organizations generate huge amount of electronic data stored in heterogeneous databases. These data are a valuable resource for mining useful knowledge to support scientific decision-making in order to improve efficiency in diagnosis and treatment of diseases. Medical knowledge mining is an emerging area of computational intelligence applied to automatically analyze electronic medical records and health databases. The non-hypothesis driven analysis approach of data mining technology can induce knowledge from clinical data repositories and health databases. Various data mining methods have been proposed [2, 3, 4, 7] to learn useful knowledge from medical data. Major techniques adopted by many researchers are rule induction and classification tree generation with the main purpose to support medical diagnosis [1, 5, 6].

Our work is also in the main stream of medical decision support system, but our methodology is different from those appeared in the literature. The knowledge-mining system proposed in this paper is based on logic programming paradigm. The justification of our logic-based system is that the closed form of Horn clauses that treats program in the same way as data facilitates fusion of knowledge learned from different sources, which is a normal setting in medical domain. Knowledge reuse can easily practice in this framework.

2 Medical knowledge-mining system: Its framework and implementation

Health information is normally distributive and heterogeneous. Hence, we design the medical knowledge-mining system (Figure 1) to include data integration component at the top level to

collect data from distributed databases and also from documents in text format. This component has been designed to input and select data with natural language processing and currently under construction. Knowledge base in our design stores both induced knowledge, in which its significance has to be evaluated by the domain expert, and background knowledge encoded from human experts. Knowledge inferring and reasoning is the module interfacing with medical practitioners and physicians at the front-end and accessing knowledge base at the back-end.

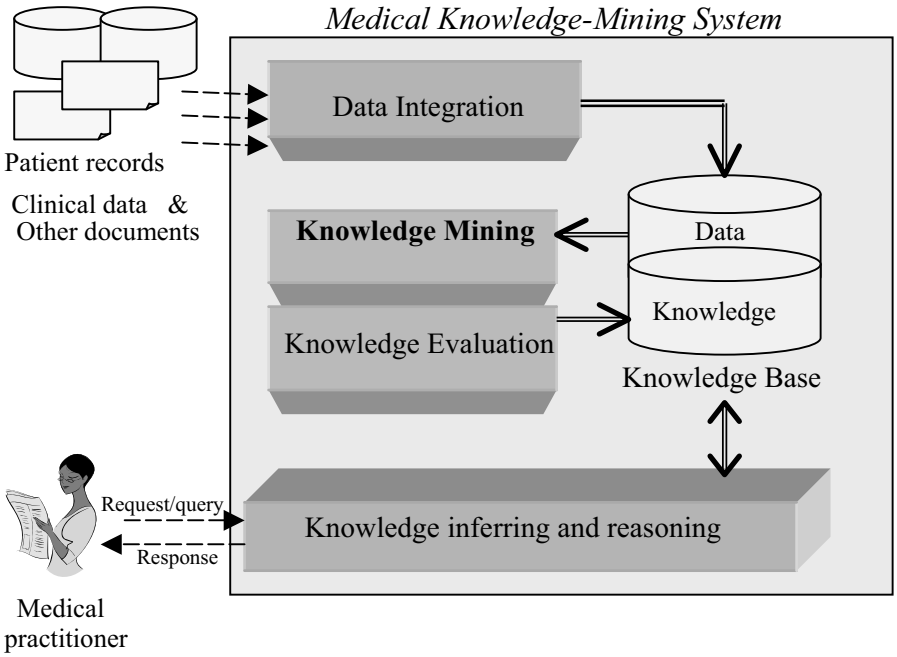


Figure 1. The framework of medical knowledge-mining system

The implementation of knowledge-mining component is based on decision-tree induction algorithm [8]. Program coding adopts the syntax of SWI prolog (www.swi-prolog.org). The main module calls initialization procedure and starts creating edges and nodes of the decision tree. The data to be used by main module to create decision tree is in another Prolog file (data.pl). It can be noticed that program and data take the same format; that is, all are in Prolog clausal form.

```
:- include('data.pl').
:- dynamic current_node/1,node/2,edge/3.

main :-  init(AllAttr,EdgeList),  getNode(N), create_edge(N,AllAttr,EdgeList), print_model.

init(AllAttr,[root-nil/PB-NB]) :-  retractall(node(_,_)), retractall(current_node(_)),
            retractall(edge(_,_,_)), assert(current_node(0)) ,
            findall(X, attribute(X,_), AllAttr1), delete(AllAttr1, class, AllAttr),
            findall(X2,instance(X2,class=home,_),PB), findall(X3,instance(X3,class=ward,_),NB).
getNode(X) :- current_node(X), X1 is X+1, retractall(current_node(_)),assert(current_node(X1)).

create_edge(_,_,[]) :- !.     create_edge(_,[],_) :- !.
create_edge(N, AllAttr, EdgeList) :-   create_nodes(N, AllAttr, EdgeList).

create_nodes(_,_,[]) :- !.     create_nodes(_,[],_) :- !.
```

```
create_nodes(N, AllAttr, [H1-H2/PB-NB|T]) :-  getNode(N1), assert(edge(N,H1-H2,N1)),
                assert(node(N1,PB-NB)),  append(PB, NB, AllInst),
                ( (PB \== [], NB \== []) ->  (cand_node(AllAttr, AllInst, AllSplit),
                        best_attribute(AllSplit,[V, MinAttr, Split]),
                        delete(AllAttr, MinAttr, Attr2),  create_edge( N1, Attr2, Split)) ; true ),
                create_nodes(N, AllAttr, T).
best_attribute([], Min, Min).
best_attribute([H|T], Min) :- best_attribute(T, H, Min).
best_attribute([H|T], Min0, Min) :-  H=[V,_, _ ], Min0 = [V0, _, _ ],
                ( V<V0 -> Min1 = H; Min1 = Min0), best_attribute(T, Min1, Min).
cand_node([],_,[]) :- !.  cand_node(_,[],[]).  % generate candidate decision node
cand_node([H|T],CurInstL,[[Val,H,SplitL]|OtherAttr]) :-  info(H, CurInstL, Val, SplitL),
                cand_node(T,CurInstL,OtherAttr).
info(A,CurInsL,R,Split) :- attribute(A,L), maplist(concat3(A,=),L,L1), suminfo(L1,CurInsL,R,Split).
concat3(A,B,C,R) :- atom_concat(A,B,R1),  atom_concat(R1,C,R).
suminfo([],_,0,[]).
suminfo([H|T], CurInstL, R, [Split | ST]) :-  AllBag=CurInstL, term_to_atom(H1,H),
        findall(X1, (instance(X1,_,L1), member(X1, CurInstL), member(H1,L1)), BagGro),
        findall(X2,(instance(X2,class=home,L2), member(X2,CurInstL), member(H1,L2)), BagPos),
        findall(X3,(instance(X3,class=ward,L3), member(X3,CurInstL), member(H1,L3)), BagNeg),
       (H11=H22) =H1, length(AllBag, Nall), length(BagGro, NGro), length(BagPos, NPos),
       length(BagNeg, NNeg), Split = H11-H22/BagPos-BagNeg,  suminfo(T, CurInstL, R1,ST),
       ( NPos is 0 *->L1 = 0;  L1 is (log(NPos/NGro)/log(2)) ),
       ( 0 is NNeg *->L2 = 0; L2 is (log(NNeg/NGro)/log(2)) ),
       ( NGro is 0 -> R= 999; R is (NGro/Nall)*(-(NPos/NGro)*L1-(NNeg/NGro)*L2)+R1 ).
```

3 Experimental results and conclusion

The proposed system was tested with the discharge-decision data of 86 patients after their operations. The decisions are either the patient is in good condition and ready to go home, or the patient seems to be fine but should be sent to the general ward for further observation and follow-up check. From the data set containing eight observed attributes, the knowledge-mining system discovers that final decision making are based on two main conditions: the stability of core temperature and the degree of patient's comfort. The running result on this data set is shown graphically as a tree (left column) and also as decision rules (right column) in Figure 2.

coreTempStability = stable | comfort = 5 ➔ home | comfort = 7 ➔ home | comfort = 10 ➔ ward | comfort = 15 ➔ ward coreTempStability = mod-stable ➔ ward coreTempStability = unstable ➔ home	*If coreTempStability=stable and comfort=(5 or 7), then prepare to go home.* *If coreTempStability=stable and comfort=(10 or 15), then send to general ward.* *If coreTempStability=mod-stable, then send to general ward.* *If coreTempStability=unstable, then prepare to go home.*

Figure 2. Decision tree and rules as the results obtain from the execution of medical knowledge-mining system

In conclusion, we propose the design and implementation of an automatic knowledge-mining tool to discover knowledge from medical data. Our design includes a complete set of tools necessary for the integration, induction, and application of the knowledge discovery process. A rapid prototyping of knowledge-mining component is based on the concept of logic programming. The logic-based programming language paradigm benefits code reuse and a high level of program abstraction. The plausible extension of our current work is to add constraints into the knowledge-mining method in order to limit the search space and respond with the most relevant knowledge to the specific application.

Acknowledgements

This research has been funded by grants from the National Research Council and the Thailand Research Fund (TRF, grant number RMU5080026). DEKD is fully supported by Suranaree University of Technology.

References

[1] C. Bojarczuk et al., "A constrained-syntax genetic programming system for discovering classification rules: Application to medical data sets", *Artificial Intelligence in Medicine*, 30, 2004, pp.27-48.

[2] S. Ghazavi and T. Liao, "Medical data mining by fuzzy modeling with selected features", *Artificial Intelligence in Medicine*, 43(3), 2008, pp.195-206.

[3] M. Huang, M. Chen, and S. Lee, "Integrating data mining with case-based reasoning for chronic diseases prognosis and diagnosis", *Expert Systems with Applications*, 32, 2007, pp.856-867.

[4] N. Hulse et al., "Towards an on-demand peer feedback system for a clinical knowledge base: A case study with order sets", *J Biomedical Informatics*, 41, 2008, pp.152-164.

[5] E. Kretschmann, W. Fleischmann, and R. Apweiler, "Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on SWISS-PROT" *Bioinformatics*, 17(10), 2001, pp.920-926.

[6] E. Mugambi et al., "Polynomial-fuzzy decision tree structures for classifying medical data", *Knowledge-Based System*, 17(2-4), 2004, pp.81-87.

[7] B. Pandey and R.B. Mishra, "Knowledge and intelligent computing system in medicine", *Computers in Biology and Medicine*, 39, 2009, pp.215-230.

[8] J.R. Quinlan, "Induction of decision trees", *Machine Learning*, 1, 1986, pp.81-106.

A Principled, Complete, and Efficient Representation of C++

Gabriel Dos Reis[1*], and Bjarne Stroustrup[2]

[1] Texas A&M University,
College Station, TX-77843, USA
gdr@cs.tamu.edu

[2] Texas A&M University,
College Station, TX-77843, USA
bs@cs.tamu.edu

Abstract

We present a systematic representation of C++, called IPR, for complete semantic analysis and semantics-based transformations. We describe the ideas and design principles that shaped the IPR. In particular, we describe how general type-based unification is key to minimal compact representation, fast type-safe traversal, and scalability. The IPR is general enough to handle real-world programs involving many translation units, archaic programming styles, and generic programming using likely C++0x extensions that affect the type system. The difficult issue of how to represent irregular (ad hoc) features in a systematic (non ad hoc) manner is among key contributions of this paper. The IPR can represent all of C++ with just slightly less than 200 node types; to compare the ISO C++ grammar has over 700 productions. Finally, we report impacts of this work on existing C++ compilers.

1 Introduction

The C++ programming language [11] is a general-purpose programming language, with bias toward system programming. It has, for the last two decades, been widely used in diverse application areas [20]. Beside traditional applications of general-purpose programming languages, it is being used in high-performance computing, embedded systems, safety-critical systems (such as, airplane controls), space explorations, etc. Consequently, the demand for static analysis and advanced semantics-based transformations of C++ programs is pressing. Dozens of analysis frameworks for C++ and for combination of C++ and code in other languages (typically C and Fortran) exist [1, 15, 16], but none handles the complete C++ language. Most are specialized to particular applications, and few (if any) can claim to both handle types and be portable across compilers.

This paper discusses a complete, efficient and direct representation of C++ implemented in C++, designed as part of a general analysis and transformation infrastructure, called *The Pivot*, being developed at Texas A&M University.

The IPR does not handle macros before their expansion in the preprocessor. With that caveat, we currently represent every C++ construct completely and directly. Note that by "completely" we mean that we capture all the type information, all the scope and overload information, and are able to reproduce input line-for-line. We capture templates (specializations and all) before instantiation — as is necessary to utilize the information represented by "concepts" [5, 10]. To be able to do this for real-world programs, we also

*Correspondence to: Postal Address(es) and Tel(fax) number(s).

handle implementation-specific extensions. We generate IPR from two compiler front ends [6, 9].

Our emphasis on completeness stems from a desire to provide a shared tool infrastructure. Complete representation of C++ is difficult, especially if one does not want to expose every irregular detail to every user. Some complexity is inherent, stemming from C++'s support of a wide range of programming styles; some is incidental, stemming from a long history of evolution under a wide range of real-world pressures; some originated in the earliest days of C. Independently of the sources of the complexity, a representation that aims to be general — aims to be a starting point for essentially every type of analysis and transformation — must cope with it. Each language feature — however obscure or advanced — not handled implies lack of support for some sub-community.

Our contribution is to engineer a small and efficient library with a regular and theoretically well-founded structure for completely representing a large irregular, real-world language. The IPR library has been developed side by side with a formalism to express the static semantics of C++.

2 Design Rules

The goals of generality directly guide the design criteria of IPR:

1. *Complete* — represents all Standard C++ constructs, but not macros before expansions, not other programming languages.
2. *General* — suitable for every kind of application, rather than targeted to a particular application area.
3. *Regular* — does not mimic C++ language irregularities; general rules are used, rather than long lists of special cases.
4. *Fully typed* — every IPR node has a type.
5. *Minimal* — its representation has no redundandent values and traversal involves no redundant dynamic indirections.
6. *Compiler neutral* — not tied to a particular compiler.
7. *Scalable* — able to handle hundreds of thousands of lines of code on common machines (such as our laptops).

Obviously, we wouldn't mind supporting languages other than C++, and a framework capable of handling systems composed out of parts written in (say) C++, C, Fortran, Java, and Python would be very useful to many. However, we do not have the resources to do that well, nor do we know if that can be done well. That is, we do not know whether it can be done without limiting the language features used in the various languages, limiting the kinds of analysis supported by the complete system, and without replicating essentially all representation node and analysis facilities for each language. These questions are beyond the scope of this paper. It should be easy to handle at least large subsets of dialects. In this context, C is a set of dialects. Most C++ implementations are de facto dialects.

Within IPR, C++ programs are represented as sets of graphs. For example, consider the declaration a function `copy`:

```
int* copy(const int* b, const int* e, int* out);
```

To represent this, we must create nodes for the various entities involved, such as types, identifiers, the function, and function parameters. parameters. Some information is implicit in the C++ syntax. For example, that declaration will occur in a scope, may overload other `copy` functions, and this `copy` may throw exceptions. The IPR makes all such information easily accessible to a user. For example, the IPR representation of `copy` contains the exception specification `throw(...)`, a link to the enclosing scope, and links to other entities called `copy` in that scope.

The types `int*` and `const int*` are both mentioned twice. The IPR library unifies nodes, so that a single node represents all `int`s in a program, and another node represents all `const int`s in a program, referring to the `int` node for its `int` part. The implication of this is that we can make claims of minimality of the size of the representation and of the number of nodes we have to traverse to gather information. It also implies that the IPR is not "just a dumb data structure": it is a program that performs several fundamental services as it creates the representation of a program. Such services would otherwise have had to be done by each user or by other services. For example, the IPR implements simple and efficient automatic garbage collection.

The design of IPR is not derived from a compiler's internal data structures. In fact, a major aim of the IPR is to be compiler independent. The representation within compilers have evolved over years to serve diverse requirements, such as error detection and reporting, code generation, providing information for debuggers and browsers, etc. The IPR has only one aim: to allow simple traversals with access to all information as needed and in a uniform way. By *simple* traversal, we mean that the complexity of a traversal is proportional to the analysis or transform performed, rather than proportional to the complexity of the source language. Because the IPR includes full type information, full overload resolution, and full understanding of template specialization, it can be generated only by a full C++ compiler. That is, the popular techniques relying on just a parser (syntax analyser or slightly enhanced syntax analyser) are insufficiently general: They don't generate sufficient information for complete representation. The IPR is designed so as to require only minimal invasion into a compiler to extract the information it needs.

The IPR is a fully-typed abstract-syntax tree. This is not the optimal data structure for every kind of analysis and transformation. It is, however, a representation from which a specialized representation (*e.g.* a flow graph or an linkage specification) can be generated far more easily than through conventional parsing or major surgery to compiler internals. In particular, we are developing a high-level flow graph representation that can be held in memory together with the AST and share type, scope, and variable information.

3 Representation

Representing C++ completely is equivalent to formalizing its static semantics. Basically, there is a one-to-one correspondence between a semantic equation and an IPR node. The IPR does not just represent the syntax of C++ entities. The IPR essentially represents a superset of C++ that is far more regular than C++. Semantic notions such as overload-sets and scopes are fundamental parts of the library and types play a central role. In fact *every* IPR entity has a type, even types. Thus, in addition to supporting type-based analysis and transformation, the IPR supports concept-based analysis and transformation.

3.1 Nodes

Here, we do not attempt to present every IPR node. Instead, we present only as much of IPR as is needed to understand the key ideas and underlying principles. Each node represents a fundamental part of C++ so that each piece of C++ code can be represented by a minimal number of nodes (and not, for example, by a number of nodes determined by a parsing strategy).

3.2 Node design

The IPR library provides users with classes to cover all aspects of Standard C++. Those classes are designed as a set of hierarchies, and can be divided into two major groups:

1. abstract classes, providing interfaces to representations
2. concrete classes, providing implementations.

The interface classes support non-mutating operations only; these operations are presented as virtual functions. Currently, traversals use the Visitor Design Pattern [8] or an iterator approach.

IPR is designed to yield information in the minimum number of indirections. Consequently, every indirection in IPR is semantically significant. That is, an indirection refers to 0, 2 or more possibilities of different kinds of information, but not 1. For if there was only 1 kind of information, that kind of information would be accessed directly. Therefore an if-statement, a switch, or an indirect function call is needed for each indirection. We use virtual function calls to implement indirections. In performance, that is equivalent to a switch plus a function call [12]. Virtual functions are preferable for simplicity, code clarity, and maintenance.

The obvious design of such class hierarchies is an elaborate lattice relying on interfaces presented as abstract virtual base classes, and implementation class hierarchies, with nice symmetry between them. This was indeed our first design. However, that led to hard-to maintain code (prone to lookup errors and problems with older compilers), overly large objects (containing the internal links needed to implement virtual base classes), and slow (due to overuse of virtual functions).

The current design (described below) relies on composition of class hierarchies from templates, minimizing the number of indirections (and thus object size), and the number of virtual function calls. To minimize the number of objects and to avoid logically unnecessary indirections, we use member variables, rather than separate objects accessed through pointers, whenever possible.

Interfaces Type expressions and classic expressions can be seen as the result of unary, or binary, or ternary node constructors. So, given suitable arguments, we need just three templates to generate every IPR node for "pure C++". In addition, we occasionally need a fourth argument to handle linkage to non-C++ code, requiring a quaternary node. For example, every binary node can be generated from this template:

```
template<class Cat = Expr,  // kind (category) of node
         class First = const Expr&,
         class Second = const Expr&>
struct Binary : Cat {
   typedef Cat Category;
```

```
    typedef First Arg1_type;
    typedef Second Arg2_type;
    virtual Arg1_type first() const = 0;
    virtual Arg2_type second() const = 0;
};
```

`Binary` is the base class for all nodes constructed with two arguments, such as an array type (node) or an addition expression (node). The first template parameter `Cat` specifies the kind (category) of the node: (classic) expression, type, statement, or declaration. The other two template parameters specify the type of arguments expected by the node constructor. Most node constructors take expression arguments, so we provide the default value `Expr`. The functions `first()` and `second()` provide generic access to data.

Note how `Binary` is derived from its first argument (`Cat`). That's how `Binary` gets its set of operations and its data members: It inherits them from its argument. This technique is called "the curiously recurring template pattern" [4] or "the Barton-Nackman trick"; it has been common for avoiding tedious repetition and unpleasant loopholes in type systems for more than a decade (it is mentioned in the ARM [7], but rarely fails to surprise). The strategy is systematically applied in the IPR library, leading to linearization of the class hierarchy (see Figure 1).

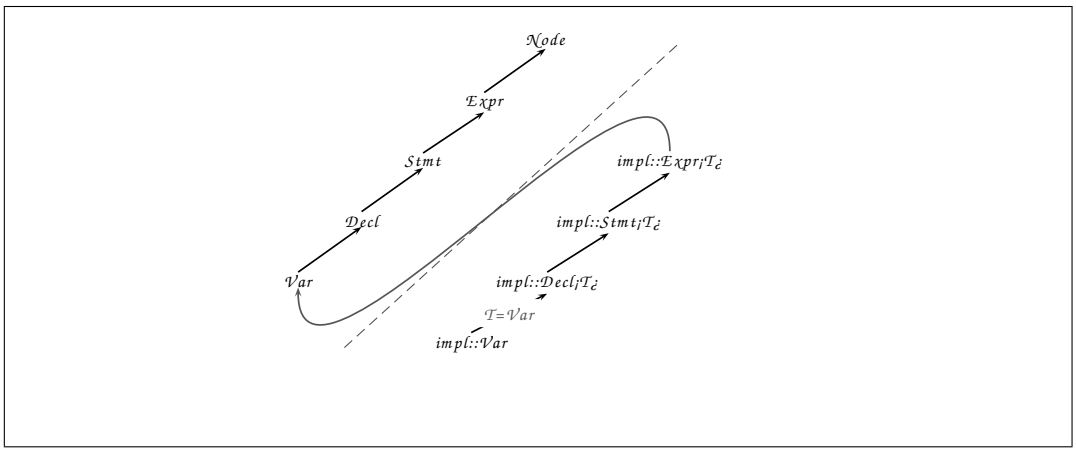


Figure 1: Current design of the IPR library

A specific interface class is then derived from the appropriate base (`Unary`, `Binary`, or `Tertiary`). For example:

```
struct Array
      : Binary<Category<array_cat,Type>, const Type&> {
   Arg1_type element_type() const { return first(); }
   Arg2_type bound() const        { return second(); }
};
```

That is, an `Array` is a `Type` taking two arguments (a `Type` and an `Expr`) and a return type (a `Type`). `Array`'s two member functions provide the obvious interface: `element_type()` returns the type of an element and `bound()` returns the number of elements. Please note that the functions `element_type()` and `bound()` are themselves *not* virtual functions; they are simple "forwarding" inline functions, therefore induce no overhead.

The category argument `Category<array_cat,Type>` exposes an implementation detail. The category is `Type` (i.e., an array is a type), but to optimize comparisons of types, we

associate an integer `array_cat` with the `Array` type. Logically, it would be better not to expose this implementation detail, but avoiding that would involve either a per-node memory overhead storing the `array_cat` value or a double dispatch in every node comparison. We introduced `array_cat` after finding node comparison to be our main performance bottleneck. So far, we have found no systematic technique for hiding `array_cat` that doesn't compromise our aim to keep the IPR minimal.

Concrete Representations Each interface class has a matching implementation class. Like the interface classes, the (concrete) implementation classes are generated from templates. In particular, `impl::Binary` is the concrete implementation corresponding to the interface `ipr::Binary`:

```
template<class Interface>
struct impl::Binary : Interface {
   typedef typename Interface::Arg1_type Arg1_type;
   typedef typename Interface::Arg2_type Arg2_type;
   struct Rep {
      Arg1_type first;
      Arg2_type second;
      Rep(Arg1_type f, Arg2_type s)
         : first(f), second(s) { }
   };
   Rep rep;

   Binary(const Rep& r) : rep(r) { }
   Binary(Arg1_type f, Arg2_type s) : rep(f, s) { }

   // Override ipr::Binary<>::first.
   Arg1_type first() const { return rep.first; }

   // Override ipr::Binary<>::second.
   Arg2_type second() const { return rep.second; }
};
```

The `impl::Binary` implementation template specifies a representation, constructors, and access functions (`first()` and `second()`) for the `Interface`. Given `impl::Binary`, we simply define `Array` as a `typedef` for the implementation type:

```
typedef impl::Binary<impl::Type<ipr::Array> > Array;
```

The `Array` type is generated as an instantiation of the `Binary` template.

3.3 Sharing

By *node sharing*, we mean that two nodes that represent the same entity shall have the same address. In particular, node sharing implies that if a node constructor is presented twice with equal lists of arguments, it will yield the same node. If node sharing is implemented for a class, that class is said to be *unified*. Since a user-defined type (classes or enums) can be defined only once in a given translation unit, sharing of nodes is suggested by C++ language rules. Every IPR node can be unified; exactly which are unified is a design choice related to performance (of the IPR itself and of applications). This can be used to tune IPR.

Implementing node sharing is easy for named types, but less straightforward for built-in types and types constructed out of other types using composition operators (e.g., `int`, `double (*)(double)`, and `vector<Shape*>`). The problem arise because such types are not introduced by declarations. They can be referred to without being explicitly introduced into a program. For example, we can say `int*` or take the address of a `double` and implicitly introduce `double*` into our set of types. Node sharing for such types implies maintenance of tables that translate arguments to constructed nodes. Since an expression does not have a name, unifying expression nodes share this problem (and its solution) with nodes for unnamed types.

We can define node sharing based on at least two distinct criteria: syntactic equivalence, or semantic equivalence. Node sharing based on syntactic equivalence has implications on the meaning of overloaded declarations; two declarations might appear overloaded even though only the spelling of their types differs. For example, the following function template declarations are possibly overloads whereas Standard C++ rules state they declare the same function.

```
template<typename T, typename U>
  void bhar(T, U);

template<typename U, typename T>
  void bhar(U, T);
```

The reason is that for templates, only the positions of template-parameters (and their kinds) are relevant. Normally, we do not care whether the name of a template-parameter is `T` or `U`; however, in real programs, people often use meaningful names, such as `ForwardIterator` instead of `T`.

3.4 Effects of unification

Space is time. It should be obvious that, because nodes are not repeatedly created to represent the same type, node sharing leads to reduced memory usage and less scattering in the address space (and therefore few cache misses.) Experiments with the classic first program

```
#include <iostream>
int main()
{
   std::cout << "Hello, World" << std::endl;
}
```

based on GCC-3.4.2 — at the time we started the IPR project — reveal that, in non-sharing mode, there are 60855 calls to type constructors; out of which we have

1. 60% for named types (only less than 1% are syntactically distinct),
2. 17% for pointer types,
3. 11% for `const`-qualified types,

Due to curiosities in the GCC compiler infrastructure, we cannot get precise counting of nodes, so the above are approximates (±5%). However, the GCC representation was about 32 times the size of the IPR representation. The "Hello, World!" program is useful because it drags in so much relatively advanced code though its `#include`. However, even

for medium-sized programs we must multiply the figures by at least 100 to get realistic measures, and then our savings in time and space begin to appear significant. Once we start to represent multiple translation units simultaneously, unification becomes a critical component of scalability.

Inspired by our design and our measurements, GCC has switched to a unified internal representation of types.

For program analysis that requires type comparison, node sharing offers time efficiency because type comparison is reduced to pointer comparison. This is significant because many forms of analysis (as well as the IPR itself) basically boil down to "traverse the program representation doing a lot of comparisons along the way to decide which nodes need attention". With node sharing, those comparisons are simple pointer comparisons. Without node sharing they are recursive double dispatch operations. Obviously, the time gained sharing nodes should be weighted against the overhead of building and using hash tables to be able to find existing nodes when you make a new node.

In the context of program transformation, another advantage of node sharing is consistency. Since there can be only one node for a type named `Foo`, we never need to walk through the whole graph to modify the properties of all `Foo`s. That is an important property when merging separately compiled translation units, doing whole-program analysis, and doing systematic substitutions. For example, with a single substitution, we can replace all uses of a type, say `int[]`, with another type, say `vector<int>`, in a whole translation unit.

4 Details

"The devil is in the details." If C++ had been designed yesterday with "simple complete representation" as a major goal, representing it would have relatively been easy. Basically, the previous section would have been the end of the story. However, elements of C++ were designed more than 30 years ago (for pre-K&R C) and much (both standard and non-standard) have been added since. This seriously complicates the design of a complete representation for C++. However, these "details" must be dealt with to produce a tool, rather than a toy. If you feel like commenting "C++ is just too complicated, let's work on tools for another language" then think what our favorite language might look like after 20 years of serious industrial use and also consider the number of people that will benefit from extra work required for a mature language.

Dealing with "details" has been much more than 50% of the total design effort. The "details" are plentiful and irregular. However, we must fit them into a more general framework so that the IPR user do not need to remember (and handle) a long list of special cases. In other words, we cannot take an ad hoc approach to dealing with ad hoc language features. We must abstract the many "details" into a few IPR constructs.

4.1 Lexical and home scopes

A name can simultaneously belong to more than one scope. For example:

```
int f(int i) {
   extern int g(int);     // g is global
   return g(i);
}
```

```
class A {
   friend void f(A) { }   // f is in enclosing scope; visible
                          // only through ‘‘argument dependent’’
                          // name lookup
};

namespace N {
   extern "C" void bar();  // bar is global
}
```

In the function `f(int)`, the locally declared function `g(int)` is visible only in the local block established by the body of `f(int)`. However, it really belongs to the global scope; that is, there is no nested or local functions in C or C++. The function `f(A)` defined in the class `A` really belongs to the enclosing namespace scope of `A`. However, an ordinary name lookup will not find it (unless a matching declaration is also available in that scope, which is not the case here). That function is visible only through a special name lookup (*argument dependent name lookup*) that considers the syntactic form of a call and the type of the arguments. The third example declares the function `bar()` as having a "C" language calling convention, consequently it really belongs to the global scope. However name lookup will not find it in the global scope – it is visible only the scope of `N`. Note also that there can be only one such function in the whole program named `bar` with that same type and "C" calling convention.

Note that the first example is also C and fairly common in C-style C++ code.

The general solution to all of these problems (and more) is that every declaration has a *lexical scope* and some also have a *home scope*. All information relating to the entity declared can be found though its entry in its home scope.

```
struct Decl : Stmt {
   // ...
   virtual const Name& name() const = 0;
   virtual const Region& home_region() const = 0;
   virtual const Region& lexical_region() const = 0;
   // ...
};
```

The lexical region is the scope in which the declaration appear in the source text. The home region the scope in which the declaration really belongs to according to the C++ rules. For most cases, those two regions are the same. However, for each of the examples above the home region and lexical region differ.

4.2 Overloading, specialization, etc.

Often, several declarations are related. For example, a function can have several declarations (which must match) and several functions in a scope can have the same name (so that they must be considered together for overload resolution). Of course, IPR must keep the information that the programmer provided (the many declarations), but it must also present a single entity (the function, the variable, the template) to the user unless the user express an interest in "the details". Consider:

```
void print(double);

void f(int i) { print(i); }
```

```
void print(int);

void g(int i) { print(i); }

void print(double d) { cout << d; }
```

The IPR represents different functions with the same name in the same scope as overload sets, but different declarations of the same function are linked to the first declaration of that function: The `Decl` class handles all linked declarations with just three functions:

```
struct Decl : Stmt {
   // ...
   virtual const Decl& master() const = 0;
   virtual const Sequence<Decl>& decl_set() const = 0;
   virtual const Decl& defining_decl() const = 0;
   // ...
};
```

The `master()` is the first declaration of a given name encountered. The `decl_set()` is the set of all declarations of that name. The `defining_decl()` is the defining declaration.

Both the primary declaration and all the secondary declarations are placed in their proper scopes and their proper places in that scope. This is essential: Note how you can change the meaning of the program fragment above by reordering the declarations. This is unfortunate, but follows directly from the C++ standard and is used in real code.

The distinction between an overload set and a linked set of declarations of the same entity also directly reflects the C++ distinction between overloading and specialization.

4.3 Lowering

Even at the level of an AST, different users want different levels of representation. We have already "lowered" the representation of the program by expanding macros, so that the IPR represents a compiler's view of a program, rather than the view of a programmer looking at a screen. This is an important design decision for IPR and not one that's always ideal. However, we don't think we had much choice. Macros are inherently irregular, so that distinctions among fundamental notions — such as, declaration, statement, and expression — are often blurred by macros.

The next major design choice is whether to retain `typedef` names. For example:

```
typedef int Length;
Length x;
```

Is `x` a `Length` or an `int`. According to C++, it's an `int` because a typedef name is only an alias. However, `Length` has a meaning to some programmer and some forms of analysis assign meaning to typedef names (and to other aliases). "Other aliases" include namespace aliases, using declarations, and (in C++0x) template aliases. It is important that the IPR implement a uniform policy vis a vis aliases.

Finally, there is the issue of how to represent member access and uses of overloaded operators. Consider:

```
void f(T x, TT p) {
     ++x;
     T(x) = 5;
     p->f();
}
```

We could represent `++x` as a use of operator ++ or as call node for the function `operator++()`. The first alternative is the user's view, the syntactic view. The latter view is "lowered" to reflect a semantic view. For example, lowering to a uniform function call notation simplifies programs concerned with program execution.

It is important to have a uniform policy on this kind of examples. Several times we (as have others) thought we had a free choice in such decisions for a specific operator, language construct, or type. In fact, we do not. Consider the case where the example above is a template function with `T` as an unconstrained template parameter. In that case, we cannot even know whether `T(x)` is a cast or a declaration of a variable `x` with redundant parentheses! Any uniform policy in a system that fully handles templates must retain the syntactic view – any lowering will be premature. Also, the syntactic view is the only one that allows re-generation of the user's code without risk of subtle semantic changes. For example, if we transformed `++x` to a uniform call syntax (say) `operator++(&x)`, we would not (without additional information) know whether the user wrote `++x` or `x.operator++()` or `operator++(&x)`.

So, to preserve information and thereby support a larger set of applications, the IPR doesn't lower by default. If you want lowering, you can ask IPR to do so at creation time. This works well, but even that may be a premature optimization and we are considering replacing the option to lower by a lowering IPR-to-IPR tool.

Note that before lowering, IPR will take a purely syntactic view of aliases. For example:

```
typedef int Length;
// ...
void f(Length);
void f(int);
```

Before lowering, the IPR – like a naive human reader – will think that there are two functions (syntactic equivalence) whereas after lowering it will realize that there really (according to C++) is only one. The distinction can be useful for some forms of analysis.

4.4 Proprietary extensions

Most compiler providers have a host of proprietary language extensions that the average end user doesn't see. However, the deep internals of most standard libraries are littered with them. Try representing the innocent-looking "Hello, world!" program:

```
#include<iostream>

int main()
{
     std::cout << "Hello, world!";
}
```

To do this, we have to handle dozens of proprietary extensions. Such extensions (of course) differ from provider to provider and it is not unusual that they vary from release to release. They tend to be plentiful in the lowest levels of code (OS interfaces, I/O, memory management, etc.), so the standard headers included to compile "Hello, world!" is a good place to look for them. For example, in `<iostream>` from GCC-4.3.0, we find five extensions in what should have been a simple one-line function declaration:

```
extern int
snprintf (char *__restrict __s, size_t __maxlen,
```

```
        __const char *__restrict __format, ...)
throw () __attribute__ ((__format__ (__printf__, 3, 4)));
```

Often, such extensions are hidden from the programmers by wrapping them in macros, but the IPR sees through that. To deal with this, we have temporarily been reduced to the "ad hockery" of simply adding IPR nodes to represent the proprietary extensions, usually one new node per extension. Given the rate of change in these extensions, this approach is not sustainable. The "Hello, world!" program is portable and by default the IPR for it should also be. The solution is to modularize the program so that we don't represent "details" of `<iostream>` in the IPR unless the user explicitly requests it. Such requests may be non-portable in the sense that an IPR implementation for a given compiler (or version of a compiler) may not be configured to precisely handle all proprietary extensions.

Please note that not handling proprietary extensions is not an option for a general representation, such as IPR, even though it can be for a specialized representation (say) aimed at the specific task of parallelizing array computations.

4.5 Separate compilation and whole-program analysis

Real-world C++ programs consist of many separately-compiled translation units. Each translation unit often consists of many hundreds of header files recursively `#include`d by a single source code file. As described so far, the IPR represents a C++ translation unit as it appears after preprocessing; that is, as a single source file with the information from the header files included and macros expanded. We can handle multiple translation units by storing the IPR for many units and then reading them back in. The ability to store the IPR in what we call XPR ("eXternal Program Representation") format is essential because most C++ compilers cannot compile two translation units in a single invocation.

The fact that IPR is unified is most useful here because that way every inconsistency between translation units is automatically caught. For example, we could try to generate IPR for a program with the two source files

```
// x.cpp
int glob;
int gfct();
```

and

```
// y.cpp
double glob;
void gfct();
```

The IPR will detect the two errors.

So, the IPR (supported by XPR) trivially supports whole-program analysis: Just add as many source files as you want and run traversals and transforms as usual. This is also the point where the compactness of nodes and the space savings from unification really pays off.

However, the situation is still not quite ideal. Considering the problems with proprietary extensions deep in implementations, we must consider an explicit approach to modularity. The IPR knows the source of every declaration (to the line number), so it is easy to tell what interface to a "module", such as `<iostream>` was really used by user code. This implies that we could represent a use of a module as the name of its header file plus the set of declaration nodes used to access it. That is, we can treat a header file as a parameterized

module. Generating that is a fairly simple IPR program, but the need to abstract from details of header files is so common that we are considering integrating it into the IPR itself.

Note that in general two uses of a "module" represented as a header file are not equivalent because macros, typedef, etc. can affect the set of definitions in the header and meaning of those definitions. We can trivially use the IPR to detect any differences or to detect any differences that matter for a given use, though.

4.6 Simplicity

One measure of simplicity is that the complete source code for IPR (excluding compiler-to-IPR generators) is just 2,500 lines of C++ (excluding comments). The code for IPR is available from the author.

5 Related and Future Work

The IPR was inspired by the *eXtented Type Information* library designed by the second author. XTI focused on the representation of the C++ type system, whereas IPR aims at the full C++ language. There are *many* projects [1, 2, 14, 15, 17, 18, 19] targeting static analysis and transformations of C++ programs. For example, CodeBoost [2, 3, 13] focuses on transformations of C++ programs, for numerical PDE solvers, written in the Sophus style. Simplicissimus [18, 19] and ROSE [17] are other projects for transforming C++ programs. Many of these systems are commercial and not documented in the literature. Few aim to handle full Standard C++, few aims at generality (as opposed to specific applications), and few aims at compiler independence. None — to our knowledge — aims at all three.

Obviously, our immediate aims include applications that test the generality and portability of the IPR and its associated tools. For example, conventional style analyzers, statistics gathering, and visualization tools. We have been able to represent the full source code FireFox. We will experiment with the use of concepts and library-specific validations, optimizations, and transformations in the domains of parallel, distributed, and embedded systems.

We plan to provide more ways of specifying traversals and transforms (such as ROSE and CodeBoost) and to work on better ways of specifying type-sensitive (incl. concept sensitive) traversals and transformations.

From the standpoint of the structure of the IPR, the most important direction of work is to get a better handle on modularity.

We will work to make the compiler to IPR generation more complete; it is already more complete that some popular compilers, but every lacking feature will cause a problem for someone. In addition we will try to interface the IPR to more compilers and handle more dialects.

6 Conclusion

Current frameworks for representing C++ are not general, complete, accessible and efficient. In this paper, we have shown how general, systematic, and simple design rules can lead to a complete, direct, and efficient representation of ISO Standard C++. In particular, we don't have to resort to ad hoc rules for program representation or low-level techniques

for completeness or efficiency. Unification helps maintain consistency, keeps our program representation compact (as required for scalability), and minimizes the cost of comparisons. To serve the widest range of applications, we use syntactic unification. Given syntactic unification, we can implement semantic unification by a simple transformation, whereas the other way around is impossible without referring back to the program source text. In addition to unification, careful and systematic node class and node class hierarchy design is necessary to minimize overhead and enable scaling.

Acknowledgements

This work was partly supported by NSF grant CCF-0702765.

References and Notes

[1] S. Amarasinghe, J. Anderson, M. Lam, and C.-W. Tseng. An overview of the SUIF compiler for scalable parallel machines. In *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, San Francisco, CA, 1995.

[2] O. Bagge. CodeBoost: A Framework for Transforming C++ Programs. Master's thesis, University of Bergen, P.O.Box 7800, N-5020 Bergen, Norway, March 2003.

[3] O. Bagge, K. Kalleberg, M. Haveraaen, and E. Visser. Design of the CodeBoost transformation system for domain-specific optimisation of C++ programs. In Dave Binkley and Paolo Tonella, editors, *Third International Workshop on Source Code Analysis and Manipulation (SCAM 2003)*, pages 65–75, Amsterdam, The Netherlands, September 2003. IEEE Computer Society Press.

[4] James O. Coplien. Curiously Recurring Template Patterns. *C++ Report*, 7(2):24–27, 1995.

[5] Gabriel Dos Reis and Bjarne Stroustrup. Specifying C++ Concepts. In *Conference Record of POPL '06: The 33th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 295–308, Charleston, South Carolina, USA, 2006.

[6] The Edison Design Group. `http://www.edg.com/`.

[7] Margaret E. Ellis and Bjarne Stroustrup. *The Annotated C++ Reference Manual*. Addison-Wesley, 1990.

[8] Erich Gamma, Richard Helm, Ralph Johson, and John Vlissides. *Design Patterns*. Addison-Wesley, 1994.

[9] GNU Compiler Collection. `http://gcc.gnu.org/`.

[10] Douglas Gregor, Jaakko Järvi, Jeremy Siek, Bjarne Stroustrup, Gabriel Dos Reis, and Andrew Lumsdaine. Concepts: Linguistic Support for Generic Programming in C++. In *OOPSLA '06: Proceedings of the 21st annual ACM SIGPLAN conference on Object-Oriented Programming Languages, Systems, and Applications*, pages 291–310, New York, NY, USA, 2006. ACM Press.

[11] International Organization for Standards. *International Standard ISO/IEC 14882. Programming Languages — C++*, 2nd edition, 2003.

[12] International Organization for Standards. *ISO/IEC PDTR 18015. Technical Report on C++ Performance*, 2003. Performance.

[13] K. Kalleberg. User-configurable, High-Level Transformations with CodeBoost. Master's thesis, University of Bergen, P.O.Box 7800, N-5020 Bergen, Norway, March 2003.

[14] Chris Lattner and Vikram Adve. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. In *CGO '04: Proceedings of the international symposium on Code generation and optimization*, page 75, Washington, DC, USA, 2004. IEEE Computer Society.

[15] Sang-Ik Lee, Troy A. Johnson, and Rudolf Eigenmann. Cetus — An Extensible Compiler Infrastructure for Source-to-Source Transformation. In *Proceedings of the 16th International Workshop on Languages and Compilers for Parallel Computing (LCPC)*, pages 539–553, October 2003.

[16] Georges C. Necula, Scott McPeak, Shree Prakash Rahul, and Westley Weimer. CIL: Intermediate Language and Tools for Analysis and Tranformations of C Programs. In *Proceedings of the 11th International Conference on Compiler Construction*, volume 2304 of *Lecture Notes in Computer Science*, pages 219–228. Springer-Verlag, 2002. `http://manju.cs.berkeley.edu/cil/`.

[17] M. Schordan and D. Quinlan. A Source-to-Source Architecture for User-Defined Optimizations. In *Proceeding of Joint Modular Languages Conference (JMLC'03)*, volume 2789 of *Lecture Notes in Computer Science*, pages 214–223. Springer-Verlag, 2003.

[18] S. Schupp, D. Gregor, D. Musser, and S.-M. Liu. User-extensible simplification — type-based optimizer generators. In R. Wihlem, editor, *International Conference on Compiler Construction*, Lecture Notes in Computer Science, 2001.

[19] S. Schupp, D. Gregor, D. Musser, and S.-M. Liu. Semantic and behavioural library transformations. *Information and Software Technology*, 44(13):797–810, October 2002.

[20] Bjarne Stroustrup. C++ Applications. `http://www.research.att.com/~bs/applications.html`.

On the Future of Computer Algebra Systems at the Threshold of 2010

Stephen M. Watt

University of Western Ontario,
London, Ontario, Canada n6a 5b7
Stephen.Watt@uwo.ca

Abstract

This paper discusses ways in which software systems for computer algebra could be improved if designed from scratch today rather than evolving designs from the 1980s.

1 Introduction

The prospect of building a general purpose computer algebra system from scratch is both daunting and exciting. On one hand, the sheer magnitude of the effort compared to the expected tangible reward is a tremendous barrier that few are able or willing to tackle. On the other hand, a blank slate entices us to consider what new generation of problems could be solved that would be difficult to address by incremental evolution of existing systems.

Computer algebra systems incorporate substantial amounts of code embodying sophisticated mathematical algorithms. Building a general purpose computer algebra system is a large effort requiring specialized human resources. Not only must the developers of a system be judicious software architects and skilled programmers, they must also have a high degree of mathematical expertise. Even if a group *could* build a new system, there is the question of whether another system *should* be built. There is considerable benefit in having a community where individuals can build on each other's efforts. Dividing the existing, relatively small community can lower this synergistic effect.

Setting aside for the moment whether we actually can or should build another major computer algebra system, we can consider the question of how a system built in 2010 would be different from the current systems, whose basic structures were for the most part established decades ago. This is the question addressed in the present article.

2 What is Different Today

Let us first take stock of the environmental factors that are different today than when the current generation of established systems were conceived. Some of these factors would affect any new design effort and are not specific to computer algebra.

Model of Interaction

The currently pervasive model of computer algebra is that of a dialogue between a user and a computer in an interactive session. In contrast to the previous batch systems, the direction of the computation can be decided by the user based on the results of each step. While this has been sufficient for many uses, we should ask what other models of interaction have proven useful in other applications.

Collaboration Today we see the emergence of social media as a common model of computer interaction. Groups collaborate or whole communities interact using a networked computing system as an intermediary, where the computational power is almost incidental. Today both pure and applied mathematics is much more collaborative endeavour than in the past decades. Natural support for technical collaboration is a new area of opportunity for our systems.

Exploration The worksheet/notebook model of interaction forces us to think in a linear fashion about a single line of computation. In solving problems, however, it is very often desirable to try several avenues of approach at the same time. In this situation one wishes to switch among cases, advancing a tree of exploration until one solution is found or perhaps all cases are completely explored. Although some earlier systems supported this [1], the popular computer algebra systems today offer essentially no assistance at case management or switching back and forth among contexts in which different assumptions hold.

Presentation Much work has been done in other areas on summarizing data usefully. Our systems for symbolic mathematical computation, however, for the most part present mathematical objects either as fully explicit expressions, sometimes taking thousands of lines, or as graphs of one sort or another. What has been discussed early on [10], but never well realized, is the presentation of values more succinctly while highlighting the aspects of interest. This direction can be developed quite a bit using programmatically identified features of interest that vary from application to application. Once we start thinking in this direction, it quickly follows that we should not see these graphical and expression presentations as different derived values, but rather as different simultaneous views of the same object.

Manipulation Users today are used to manipulating objects directly in a visual setting. Direct manipulation is a natural paradigm for mathematical expression transformation, but little has been done in this area. Our current systems provide many operations for transforming expressions by applying various identities or sophisticated algorithms. What has been lacking, however, in most of our systems is the ability to work on *subexpressions* in a similar manner. Direct manipulation of subexpressions, applying identities or transformations in place, can give a qualitatively different style of interaction. Being able to perform direct manipulation through multiple views leads to many interesting possibilities.

Input modalities Modern user interfaces are making use of a broad range of modalities, from the usual keyboard and mouse, to voice, cameras and various motion capture devices. It is tempting to let the imagination run wild here, but there are some very practical and obvious next steps. One is the digital pen that is now commonly available on Tablet PCs, digital white boards and PDAs [6]. Not only would it be natural to enter equations using handwritten two-dimensional notation, but there are a number of gestures commonly used in simplification. These include canceling or combining terms in a sum or factors in a quotient. A second use of the digital pen would be for sketching or making annotations. Personally, I find that when I am working on a problem I almost always make use of various informal *ad hoc* notations as tool for thought. Using a digital pen here would allow this thought process to flow naturally, without the distracting mechanics of some drawing program, plus it would be useful to keep these notes together with the computation.

Locus

Related to the model of interaction is the question of locus of code and data, not only in support of collaboration but also in support of individuals with multiple computing resources.

Data It is increasingly uncommon for applications or data to be confined to a single device. It has been commonplace for more than a decade to access information via various network protocols, but more recently it has become usual to update data in a universal store this way. This is one of the principal ingredients of cloud computing and opens many opportunities for symbolic mathematical computing, for example evolving shared databases of mathematical definitions, facts, proofs and constructions as well as objectives and conjectures. As these build on each other, interface mechanisms will be required to ensure the correctness of the compositions.

Programs Mathematical software has the fortunate property that, compared to other applications, it is relatively easy to specify cleanly what a program is supposed to do. There is therefore the possibility to have a variety of components to solve the same problem. This could include versions of code maintained by experts deploying from their own servers. It also allows, for example, simple versions that can be deployed freely and rapidly, sophisticated versions using different algorithms for higher performance, versions generating results carrying correctness certificates or domains of applicability, and so on.

Access Universal store need not be shared among different users — it may also be used by a single user to provide access to his or her data and computations from various locations and devices. User interface issues arise in how to create, explore and manipulate mathematical objects from a wide variety of devices, including personal workstations, tablet PCs, smart phones and digital whiteboards.

Computation Modern server farms use virtualization extensively to deploy computational resources flexibly, and grid computing has become a practice to solve scientific and technical problems, principally so far of a numerical nature. Locus of computation may also be determined by the location of specialized web services, where the application code must run at a specific location for technical or economic reasons.

Embedding At the other end of the spectrum, we have increased opportunity for embedded computer algebra in a wider range of applications than ever before. For example, document processing software was earlier dwarfed by the size of computer algebra systems. But now these systems can be very sophisticated, with lexical and grammatical knowledge of many languages, advanced multilingual formatting and so on. Computer algebra software for reformatting mathematical expressions intelligently can now exist as a small component of such document editors. Optimizing compilers are another example where an embedded computer algebra component would be relatively small compared to the overall system. Such a component would be useful in reformulating code to make use of identities, to share non-obvious common sub-expressions or to solve optimization sub-problems. At a lower level, computer algebra has long been used in the design of devices (such as error-correcting disk controllers), but we now have the possibility to compute symbolic or symbolic-numeric values on the fly.

Computing Power

Present computer algebra systems had their basic design decided in an era when it was envisaged they would run in an environment with orders of magnitude less memory and processor cycles. This changes several design points.

We have more computing power than ever before... Today's computers have more cache memory than there was primary memory in the design for today's most used computer algebra systems. The speed of single processors has likewise scaled up. We must therefore review all the underlying assumptions in our system designs, from data structures to patterns of memory access. In some situations we have overly complicated approaches to problems that can be greatly simplified in today's more powerful computing environments. In others, we have methods that were perfectly acceptable for smaller systems, but cause significant inefficiencies with today's architectures. For example, the order in which memory should be traversed in garbage collection is greatly dependent on the specifics of the memory hierarchy.

... and it is still not enough In the initial design of our current computer algebra systems, it was possible to consider that problems of a size that would fit in memory were of a size suitable for classical algorithms. Now, any general purpose system must consider that problems that can be handled by classical methods are not the principal bottleneck. At the same time, multi-core processors with highly parallel graphical processing units are the norm for personal computers. We are at a stage where high performance computer algebra requires both asymptotically fast algorithms and taking proper advantage of modern parallel hardware.

3 Aspects of Computer Mathematics

There has always been the need to consider how computer algebra systems should interact with traditional numerical computing and graphics software. Today additional interactions should be considered.

Symbolic-numeric computation The past decade and a half has seen significant advance in our understanding of symbolic-numeric algorithms, particularly for polynomials. However neither the data structures nor the overall logic of present computer algebra libraries are organized to have symbolic-numeric objects as first-class objects that are pervasively understood. Pervasive incorporation of symbolic-numeric structures and algorithms is an important direction in providing consistent handling of algebraic objects with approximate coefficients or partially evaluated expressions on floating point data.

Specialized kernels While considering interfaces that generalize the interactions of our systems, we must also make them work well in important specialized settings. There are a variety of settings where particularly efficient special-purpose software packages are or will become available. Notably, efficient specialized packages exist for linear algebra over various fields, semi-algebraic geometry, polynomial system solving and computational group theory (*e.g.* [3, 4, 9]).

Symbolic mathematical computation The past decades have seen an increasing separation between "symbolic mathematical computation", by which I mean computation on expressions in term algebras, and "computer algebra", by which I mean algebraic algorithms in specific domains (that might involve symbols). With a few exceptions, there has been little attention to solving problems where symbols are other than variables or coefficient parameters in rational functions. The problems of polynomials with symbolic exponents or matrices with internal structure of symbolic size have been considered elsewhere [5, 7]. But this is a much more general problem. A systematic approach is required to handle expressions involving symbols representing unspecified objects of different types. A common conceptual framework should provide, for example, simplification of expressions involving symbolic matrices (*e.g.* $\mathbf{A}\mathbf{A}^T/\det \mathbf{A}$), polynomials with symbolic exponents, expressions involving Bessel functions of symbolic complex index, *etc.* This should be determined automatically by the algebraic specification of the domain of concrete values and should allow for partial evaluation of symbols to these values. In most cases algorithms on the symbolic values will be different than algorithms on the concrete values, completely analogously to the case with symbolic-numeric computation.

Inter-operation with proof assistants With a few notable exceptions, computer algebra systems and automated proof assistants have existed in separate circles so far. The present generation of widely used computer algebra systems has little ability to make use of mathematical facts provided by proof assistants. In the 1980s, it was arguably reasonable to take this direction based on the state of proof systems then. For example, in the design of Axiom, it was considered whether to require a complete set of axiomatic properties in the signatures of domains. This idea was rejected because, at the time, the state of the art would allow little use to be made of these properties, even to the extend of verifying their consistency. Today we should consider as a standard feature much closer interaction between proof assistance and computer algebra software. Several areas can benefit from this, including specification of interfaces among components, certification of results and domains of applicability, justification of optimizations and, in the other direction, use of efficient algebra in proofs.

Knowledge management In addition to each system managing knowledge about its own library, we can foresee that mathematical knowledge will more generally be indexed and searchable in various ways. At the moment, some systems provide rudimentary access to a mathematical dictionary. In the future, when working on mathematical objects, it should be possible to search for known facts about these objects. Initially, these facts will not be in a form that can be used directly by the software system, but rather will be for the user's benefit. So how our mathematical software system organizes and represents knowledge becomes important not only for self-organization, but also for pulling in useful information from external sources.

Longitudinal inter-operation We need to plan for longevity of our mathematical software systems. All of our currently popular systems contain code older than many of their users. (Some of the code I have written in Maple is now almost 30 years old.) This longevity has many implications, foremost among them relating to the overall system architecture. Code will be written at different times and in different places, yet still be related in terms of the objects handled. Old and new code will need to be used together in some composable manner. Old code will need to be used in unanticipated new ways.

4 Elements of a Computer Algebra System

What do these desiderata imply for the structure of future computer algebra systems? While there are many possible directions, a few things seem to be clear in any future scenario.

Modular architecture All successful computer algebra systems have a significant code base, including the core system and additional libraries. To be scalable, well-defined and well-structured interactions among the parts is important. How these interactions are structured can be designed around the criteria discussed. Systems such as Axiom and Magma have used modern algebra together with data abstraction for modularity. These algebraic ideas remain useful, but must be augmented with systematic interfaces to dovetail expression (term algebra) views of component objects. Supporting these interfaces pervasively across modules will be required to support composite mathematical types well.

Interfaces as mathematical objects In a system for symbolic mathematical computation the interface specifications can themselves be mathematical objects. These can be reasoned with to perform module selection, to form simplification rules over the appropriate equational theories, *etc.* In particular, it would be desirable to have symbolic expression algebra views generated automatically from the signatures of the algebraic interfaces.

Federation We should strive to have sufficient precision in the interfaces to mathematical modules to allow correct inter-operation of components from a variety of sources, both free and proprietary, local and remote. There should be no technological impediment to making components as low-level/efficient or high-level/abstract as desired. As discussed elsewhere, the programming languages for library development and for top-level scripting have different requirements and should therefore probably be different [8].

Library reflection A large, mature system or federation of systems will work on a vast selection of objects with a plethora of available functionality, applicable in a variety of different contexts. No single person will be able to keep track of everything offered. In our setting, the majority of the functionality will be of a mathematical nature, so the properties of the modules can themselves mathematical objects that can be manipulated, reasoned with, organized and searched. At the very least, it should be possible to provide intelligent library browsing tools based on this mathematical structure, and it is not too much to imagine that object composition and algorithm selection could be at least partially automated.

Levels of abstraction Much of mathematics is about abstraction. We may wish to work at one level of abstraction, or at several levels. Some would argue that computer algebra works precisely because we have theorems and constructions that relate these different levels. In any case, we must be able to express ideas and compute with objects at the various levels and move among them. A framework for smooth movement between elements of symbolic domains and value domains is important to allow modular development of systems. Being able to view terms as values in domains and domain elements as parts of terms should allow composition of abstractions without special extra machinery. For example, we should be able to use the same mechanisms to work with matrices of numbers, matrices of parameterized rational functions, symbolic matrix expressions and expressions involving the rings and fields themselves.

Certification Our systems will need to make some sort of statements about the correctness of their results. This arises from two directions: First, if an engineer relies on a calculation performed by a computer algebra system as part of a design, then in signing off on that design it is required to record the justification. This is jurisdiction-dependent legal issue. From a systems software point of view, the minimum requirement would be for the computation to keep track of the conditions under which it is valid (for example, which quantities have appeared as denominators and so must be non-zero). Secondly, as computer algebra systems and proof systems interact more closely, there is an increasing set of libraries that are proven correct. A different kind of correctness certificate comes from using only libraries that have been formally proven.

Computational interfaces We have discussed the need to express well-defined interfaces among components, and the need to manipulate those interfaces themselves as mathematical objects. More work is needed in this area. We will also need to transmit data between components, including a variety of modules from different sources, services at different locations and displays with different views. It may be that specific subsets of OpenMath and MathML will be sufficient for these purposes. Or it may be that some other specification language will be required. In any case, attention to these computational interfaces is required. It is here that the design will either succeed or fail in supporting separately developed modules to hang together nicely.

Human interfaces We have also discussed at length above how users' interactions with systems can evolve. Ideally, our interfaces should support all of the ideas discussed: collaboration, exploration, presentation, manipulation and input modalities. Most immediately, however, an interface that supports collaborative exploration might have the highest impact.

5 The Great Unknown

Much of the success of modern symbolic computing systems is based on powerful algebraic constructions. But this captures only a small part of what mathematicians do, and an even smaller part of how mathematics is applied in its various settings. If we are to compose a next generation system based solely on some elegant ideas of modularity, composability and reflection, we run the risk of building a system exclusively for a small set of constructive pure mathematicians.

A lot of what a symbolic computing system must do cannot be expressed readily in terms of abstract algebra alone. We must acknowledge, and provide support for, exploratory procedures where the mathematical computing is done as part of an investigation and the form the answer should take, or even the precise nature of the question, is not known in advance. Hypotheses may be made and discarded, *ad hoc* approximations may be used without justification, and a result may depend on some analytic or numeric properties. At the same time, the backbone of the system and the majority of its components must work in precisely defined ways.

There has been much discussion about symbols and unknowns in computer algebra (*e.g.* [2]). Previous systems have confounded the notions of programming variables, indeterminates and parameters in algebraic structures, universally or existentially quantified symbols, unification variables and other ideas. We really must be clearer in what is meant by sym-

bols. Our hypothetical future system must be able to deal with unknowns that arise in these and other ways.

As well as using symbols to represent values quantified over given types, we must be able to work with symbols representing values from unknown types. That is, where we do not yet know the structure for which they represent elements. Not only must we be able to compose well-defined structures that are fully understood, we must also be able to work with partially defined structures and partially thought out ideas in a contained way.

6 Conclusions

This article has given a personal view of some desirable directions for the evolution of computer algebra systems. We have not concentrated on current concerns of identifying important algebraic algorithms or high performance computing issues. These are already the subject of much fruitful work. Instead, we have focused on how computer algebra systems might be organized in the future. Certain of the points may by obvious to a practitioner in the field and others may be seen as controversial. Some of these ideas can be retro-fit to existing systems and others may have to await a new generation of systems cut from whole cloth.

Acknowledgements

I would like to thank Jacques Carette, Bruce Char and James Davenport for thoughtful comments on an earlier draft.

References and Notes

[1] A. Bonadio. Theorist (a computer program), Prescience Corp., 939 Howard St., San Francisco, CA 94103, USA, 1989.

[2] J. Davenport and Ch. Faure. The "Unknown" in Computer Algebra. Programmirovanie, Jan, 1994. 4-10.

[3] J.-Ch. Faugère, GB (software library), 2009. `http://www-calfor.lip6.fr/~jcf/Software/Gb/`

[4] J.-G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B.D. Saunders, W.J. Turner and G. Villard. LinBox: A generic library for exact linear algebra. Proc. International Congress on Mathematical Software (ICMS), World Scientific, 2002. 40-50.

[5] A.P. Sexton, V. Sorge and S.M. Watt. Reasoning with Generic Cases in the Arithmetic of Abstract Matrices, Proc. Conferences on Intelligent Computer Mathematics (CICM), Springer Verlag LNAI 5625, 2009. 138-153.

[6] E. Smirnova and S.M. Watt. Communicating Mathematics via Pen-Based Computer Interfaces. Proc. 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), IEEE Computer Society, 2008. 9-18.

[7] S.M. Watt. Two Families of Algorithms for Symbolic Polynomials. Computer Algebra 2006: Latest Advances in Symbolic Algorithms – Proceedings of the Waterloo Workshop, I. Kotsireas, E. Zima (editors), World Scientific, 2007. 193-210.

[8] S.M. Watt. What Happened to Languages for Symbolic Mathematical Computation? Proc. Programming Languages for Mechanized Mathematics (PLMMS), J. Carette and F. Wiedijk (editors), http://www.risc.uni-linz.ac.at/publications/download/risc_3120/PLMMS_proc.pdf, RISC-Linz, 2007. 81-90.

[9] V. Shoup, NTL (software library), 2009. http://www.shoup.net/ntl

[10] D. Stoutemyer. Qualitative analysis of mathematical expressions using computer symbolic mathematics. Proc. Symposium on Symbolic and Algebraic Manipulation (SYMSAC). ACM, 1976. 97–104.

Automated induction of frequent patterns with knowledge-based software engineering

Kittisak Kerdprasop and Nittaya Kerdprasop

Data Engineering and Knowledge Discovery (DEKD) Research Unit
School of Computer Engineering, Suranaree University of Technology
Nakhon Ratchasima 30000 Thailand
kittisakThailand@gmail.com, nittaya@sut.ac.th

Abstract

Frequent patterns refer to the occurrences of some data items frequently found together in the database. Automated induction of frequent patterns, also known as frequent pattern mining, is among major research topics in the area of data mining for which the efficient and effective mining techniques have been sought. In this paper, we study the problem of frequent pattern mining within the context of knowledge-based software engineering. The term knowledge-based software engineering has emerged as a cross discipline of software engineering and artificial intelligence to solve a crisis of software productivity due to difficulties associated with the engineering of complex software systems. We propose a knowledge discovery tool to capture frequent patterns in the educational data sets. Our automated tool was implemented with a higher-order logic-programming scheme so that the knowledge-intensive tasks can be efficiently coded. The implementation demonstrated in this paper can also be extended without much effort to support knowledge caption and representation in order to automatically generate knowledge content in the knowledge-base system.

1 Introduction

Frequent-pattern mining is the discovery of relationships or correlations between items in a database. Let $I = \{i_1, i_2, i_3, ..., i_m\}$ be a set of m items and $DB = \{C_1, C_2, C_3, ..., C_n\}$ be a database of n instances and each data instance contains items in the set I. A *pattern* is a set of items that occur in a data instance. The number of items in a pattern is called the length of the pattern. To search for all valid patterns of length 1 up to m in large database is computational expensive. For a set I of m different items, the search space for all distinct patterns can be as huge as 2^m-1. To reduce the size of the search space, the *support* measurement has been introduced [1]. The function *support*(P) of a pattern P is defined as a number of instances in DB containing P. Thus, *support*(P) = $|\{T \mid T \in \text{DB}, \ P \subseteq T\}|$. A pattern P is called *frequent pattern* if the support value of P is not less than a predefined minimum support threshold *minS*. It is the *minS* constraints that help reducing the computational complexity of frequent pattern generation. The *minS* metric has an anti-monotone property and is applied as a basis for reducing search space of mining frequent patterns in the well-known algorithm Apriori [1].

A logic-based approach to the development of knowledge discovery system has long been an interesting research topic among data mining and machine learning researchers. For the classification task, tree-based concept induction [4, 8] and rule induction [5] are major approaches normally adopted. Frequent-pattern mining task was mostly based on the well-known APRIORI algorithm [1]. WARMR system [2] upgraded APRIORI algorithm to discover frequent patterns. Its extension [3] was developed to discover frequent Datalog patterns and relational association rules. Our implementation approach is also based on the mathematical logic concepts, but we extend the predicate terms to the level of higher-order logic.

2 Higher-order logic programming

In logic programming, a clause is a disjunction of literals (atomic symbols or their negations) such as $p \vee q$ and $\neg p \vee r$. A statement is in clausal form if it is a conjunction of clauses such as $(p \vee q) \wedge (\neg p \vee r)$. Logic programming is a subset of first order logic in which clauses are restricted to Horn clauses. A Horn clause, named after the logician Alfred Horn [7], is a clause that contains at most one positive literal such as $\neg p \vee \neg q \vee r$. Horn clauses are widely used in logic programming because their satisfiability property can be solved by resolution algorithm (an inference method for checking whether the formula can be evaluated to true).

A Horn clause with no positive literal, such as $\neg p \vee \neg q$, which is equivalent to $\neg (p \wedge q)$, is called *query* in Prolog and can be interpreted as '*:- p, q'* in which its value (true/false) to be proven by resolution method. A clause that contains exactly one positive literal such as r is called a *fact* representing a true statement, written in clausal form as '*r :-'* in which the condition part is empty and that means r is unconditionally true. Therefore, facts are used to represent data. A Horn clause that contains one positive literal and one or more negative literals such as $\neg p \vee \neg q \vee r$ is called a *definite clause* and such clause can equivalently written as $(p \wedge q) \rightarrow r$ which in turn can be represented as a Prolog *rule* as *r :- p, q*. The symbol '*:-'* is intended to mean '$\leftarrow$', which is implication in first-order logic, and the symbol '*,*' represents the operator $\wedge$ (or 'AND'). In Prolog, rules are used to define procedures and a Prolog program is normally composed of facts and rules. Running a Prolog program is nothing more than posing queries to obtain true/false answers. The symbols p, q, r are called *predicates* in first-order logic programming and they can be quantified over variables such as *r(X) :- p(X,Y), q(Y)*. This clause has the same meaning as $\forall X (p(X,Y) \wedge q(Y) \rightarrow r(X))$.The scope of variables is within a clause. Horn clauses are thus the fundamental concept of logic programming.

Higher-order predicate is a predicate that can quantify over other predicate symbols [6]. As an example, besides the rule *r(X):- p(X,Y), q(Y)*, if we are also given the following five Horn clauses (or facts): *p(1, 2). p(1, 3). p(5, 4). q(2). q(4).* Then by asking the query: *?- r(X)*, we will get the response as '*true*' and also the first instantiation as *X=1*. If we want to know all instantiations that make *r(X)* true, we may ask the query: *?- findall(X, r(X), Answer).* We will get the response: *Answer = [1, 5]*, which is a set of all answers obtained from the predicate *r(X)* according to the given facts. The predicate symbol *findall* quantifies over the variables *X*, *Answer*, and the predicate *r*. The predicate *findall* is thus called a higher-order predicate.

3 Frequent pattern mining with higher-order logic

We implemented the frequent-pattern mining program (Figure 1) based on the APRIORI algorithm [1]. Main predicate of this program is *frequent_pattern_mining*. Upon invocation, this predicate will obtain input data, also in the format of Prolog program, from the predicate *input(Data)*. The predicate *minS(V)* specifies the minimum value of support metric. Then the main predicate starts the automated induction process by searching candidate item sets and large item sets of length one, two, three, and so forth (through the predicates *makeC1*, *makeL*, and *apriori_loop*, respectively. All highlighted terms in Figure 1 are higher-order predicates. These predicates are *maplist*, *include*, and *setof*.

The program execution results on educational data sets are shown in Figure 2 (only the five patterns with highest support values are shown). These data sets are the dropout data at the secondary level (7^{th} grade – 12^{th} grade) in the school year 2002-04 reported by California state education agencies [9]. Each data instance is a report from each school district containing 15 attributes: Locale (location of the school), LO-offered (the lowest grade of the school), HI-offered (the highest grade offered by the school), the total number of student enrollments in grade 7^{th} through 12^{th}, and the total number of dropouts at grade 7^{th} through 12^{th}.

```
frequent_pattern__mining :-  input(Data), minS(V),  makeC1(C),  makeL(C,L),
apriori_loop(L,1).
apriori_loop(L,N) :- length(L) is 1,!.
apriori_loop(L,N) :- N1 is N+1, makeC(N1,L,C), makeL(C, Res), apriori_loop(Res, N1).
makeC1(Ans) :- input(D), allComb(1, ItemSet, Ans2),  maplist( countSS(D), Ans2, Ans).
makeC(N, ItemSet, Ans) :- input(D),  allComb(2, ItemSet, Ans1),
        maplist(flatten, Ans1, Ans2),  maplist(list_to_ord_set, Ans2, Ans3) ,
        list_to_set(Ans3,Ans4),  include(len(N),Ans4,Ans5), maplist(countSS(D),Ans5,Ans).
  %scan database to find: List+N
makeL(C, Res) :- include(filter, C, Ans), maplist(head, Ans, Res).
filter(_+N) :-  input(A), length(A, I),  min_support(V), N>=(V/100)*I.
head(H+_, H).
   % arbitrary subset of the set containing given number of elements
comb(0, _, []).
comb(N, [X|T], [X|Comb]) :- N> 0, N1 is N-1,  comb(N1, T, Comb).
comb(N, [_|T], Comb) :- N> 0, comb(N, T, Comb).
allComb(N, I, Ans) :- setof( L, comb(N, I, L), Ans).
countSubset(A, [], 0).
countSubset(A, [B|X], N) :- not(subset(A, B)),  countSubset(A, X, N).
countSubset(A, [B|X], N) :- subset(A, B),  countSubset(A, X, N1), N is N1+1.
countSS(SL, S, S+N) :- countSubset(S, SL, N).
len(N, X) :- length(X, N1), N is N1.
```

Figure 1. Frequent-pattern mining logic program implemented with higher-order predicates

School year 2002 (390 data instances)	LO-offered=KG & HI-offered=12	support=0.84
	HI-offered=12 & Grade-7-dropouts=0	support=0.76
	HI-offered=12 & Grade-8-dropouts=0	support=0.74
	LO-offered=KG & Grade-7-dropouts=0	support=0.74
	Grade-7-dropouts=0 & Grade-8-dropouts=0	support=0.71
School year 2003 (565 data instances)	LO-offered=KG & Grade-7-enrollment=[1-1000]	support=0.67
	LO-offered=KG & Grade-8-enrollment=[1-1000]	support=0.67
	Grade-7-enrollment=[1-1000] & Grade-8-enrollment=[1-1000]	support=0.66
	Grade-7-dropouts=0 & Grade-8-dropouts=0	support=0.65
	LO-offered=KG & Grade-7-dropouts=0	support=0.63
School year 2004 (555 data instances)	LO-offered=KG & Grade-7-enrollment=[1-1000]	support=0.67
	Grade-7-enrollment=[1-1000] & Grade-8-enrollment=[1-1000]	support=0.67
	LO-offered=KG & Grade-8-enrollment=[1-1000]	support=0.66
	LO-offered=KG & HI-offered=12	support=0.59
	Grade-7-dropouts=0 & Grade-8-dropouts=0	support=0.59

Figure 2. Running results of frequent-pattern mining program on data sets of California school dropouts during the school years 2002-04

4 Conclusions

Logic programming is a declarative style of writing programs. It is a very high-level language in the sense that it focuses on the computation's logic rather than the mechanics. Logic programming languages such as Prolog are admittedly suitable for rapid prototyping of new ideas or new program architecture. This programming paradigm is based on a solid foundation of first-order logic in which the logical properties of a computation and the coverage of predicates and quantification are emphasized. First-order logic, however, poses a restriction on the type of variables appearing in quantification not to include predicates. Higher-order logic, on the contrary, allows variables to quantify over predicates. With such relaxation, higher-order logic facilitates the implementation of a knowledge-intensive application such as frequent pattern mining, which is the discovery of frequent recurring correlations between data items in the database.

In this paper, we demonstrated a technique to implement frequent-pattern mining algorithm with the higher-order logic concept. The running examples were also illustrated through the educational data sets in which strong patterns of California state school dropout students during the year 2002-04 had been discovered. The pattern-mining program presented in this paper can also be applied to data sets in other domains.

Acknowledgements

This research has been funded by grants from the National Research Council of Thailand (NRCT) and the first author is supported by the Thailand Research Fund (TRF, grant number RMU5080026). DEKD has been fully supported by Suranaree University of Technology.

References

[1] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. (1996). Fast discovery of association rules. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining*, pp.307-328. AAAI Press.

[2] Dehaspe, L. & Toivonen, H. (1999). Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1): 7-36.

[3] Dehaspe, L. & Toivonen, H. (2001). Discovery of relational association rules. In S. Dzeroski and N. Lavrac (Eds.), *Relational Data Mining*, pp.189-212. Springer.

[4] Kramer, S. & Widmer, G. (2001). Inducing classification and regression trees in first order logic. In S. Dzeroski and N. Lavrac (Eds.), *Relational Data Mining*, pp.140-159. Springer.

[5] Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, 13:245-286.

[6] Naish, L. (1996). Higher-order logic programming in Prolog. *Technical Report 96/2*, Department of Computer Science, University of Melbourne, Australia.

[7] Nienhuys-Cheng, S.-H. & Wolf, R. (1997). *Foundations of Inductive Logic Programming*. Springer.

[8] Quinlan, J. & Cameron-Jones, R. (1993). FOIL: A midterm report. *Proceedings of the 6th European Conference on Machine Learning*, LNAI 667, pp.3-20. Springer.

[9] Sable, J, & Gaviola, N. (2007). *NCES Common Core of Data Local Education Agency-Level Public-Use Data File on Public School Dropouts: School Year 2002-04*. National Center for Education Statistics, Institue of Education Sciences, U.S. Department of Education, Washington, D.C. (http://nces.ed.gov/pubsearch/)

Author Index

MIレクチャーノートシリーズ刊行にあたり

本レクチャーノートシリーズは、文部科学省21COEプログラム「機能数理学の構築と展開」(H.15-19年度) において作成したCOE Lecture Notes の続刊である。今後、レクチャーノートは、文部科学省大学院教育改革支援プログラム「産業界が求める数学博士と新修士養成」(H19-21年度) および、新しく採択された同グローバルCOEプログラム「マス・フォア・インダストリ教育研究拠点」(H.20-24年度) の推進において招聘する国内外の研究者による講義の講義録として出版するものである。

平成20年7月
グローバルCOEプログラム
マス・フォア・インダストリ教育研究拠点
拠点リーダー　若山正人

The Joint Conference of ASCM 2009 and MACIS 2009

Asian Symposium on Computer Mathematics
Mathematical Aspects of Computer and Information Sciences

発　行　2009年12月14日
編　集　九州大学大学院数理学研究院
発　行　九州大学大学院数理学研究院
グローバルCOEプログラム「マス・フォア・インダストリ教育研究拠点」
大学院教育改革支援プログラム「産業技術が求める数学博士と新修士養成」
九州大学産業技術数理研究センター
〒819-0395 福岡市西区元岡744
九州大学伊都キャンパス数理学研究教育棟 GCOE事務室
TEL 092-802-4404　FAX 092-802-4405
URL http://gcoe-mi.jp/

印　刷　城島印刷株式会社
〒810-0012 福岡市中央区白金２丁目９番６号
TEL 092-531-7102　FAX 092-524-4411

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note	Mitsuhiro T. NAKAO Kazuhiro YOKOYAMA	Computer Assisted Proofs - Numeric and Symbolic Approaches - 199pages	August 22, 2006
COE Lecture Note	M.J.Shai HARAN	Arithmetical Investigations - Representation theory, Orthogonal polynomials and Quantum interpolations- 174pages	August 22, 2006
COE Lecture Note Vol.3	Michal BENES Masato KIMURA Tatsuyuki NAKAKI	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2005 155pages	October 13, 2006
COE Lecture Note Vol.4	宮田　健治	辺要素有限要素法による磁界解析-機能数理学特別講義 21pages	May 15, 2007
COE Lecture Note Vol.5	Francois APERY	Univariate Elimination Subresultants - Bezout formula, Laurent series and vanishing conditions - 89pages	September 25, 2007
COE Lecture Note Vol.6	Michal BENES Masato KIMURA Tatsuyuki NAKAKI	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2006 209pages	October 12, 2007
COE Lecture Note Vol.7	若山　正人 中尾　充宏	九州大学産業技術数理研究センター キックオフミーティング 138pages	October 15, 2007
COE Lecture Note Vol.8	Alberto PARMEGGIANI	Introduction to the Spectral Theory of Non-Commutative Harmonic Oscillators 233pages	January 31, 2008
COE Lecture Note Vol.9	Michael I. TRIBELSKY	Introduction to Mathematical modeling 23pages	February 15, 2008
COE Lecture Note Vol.10	Jacques FARAUT	Infinite Dimensional Spherical Analysis 74pages	March 14, 2008
COE Lecture Note Vol.11	Gerrit van DIJK	Gelfand Pairs And Beyond 60pages	August 25, 2008
COE Lecture Note Vol.12	Faculty of Mathematics, Kyushu University	Consortium “MATH for INDUSTRY” First Forum 87pages	September 16, 2008
COE Lecture Note Vol.13	九州大学大学院 数理学研究院	プロシーディング「損保数理に現れる確率モデル」 ― 日新火災・九州大学 共同研究2008年11月 研究会 ― 82pages	February 6, 2009

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.14	Michal Beneš, Tohru Tsujikawa Shigetoshi Yazaki	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2008 77pages	February 12, 2009
COE Lecture Note Vol.15	Faculty of Mathematics, Kyushu University	International Workshop on Verified Computations and Related Topics 129pages	February 23, 2009
COE Lecture Note Vol.16	Alexander Samokhin	Volume Integral Equation Method in Problems of Mathematical Physics　50pages	February 24, 2009
COE Lecture Note Vol.17	矢嶋　　徹 及川　正行 梶原　健司 辻　　英一 福本　康秀	非線形波動の数理と物理　66pages	February 27, 2009
COE Lecture Note Vol.18	Tim Hoffmann	Discrete Differential Geometry of Curves and Surfaces　75pages	April 21, 2009
COE Lecture Note Vol.19	Ichiro Suzuki	The Pattern Formation Problem for Autonomous Mobile Robots ―Special Lecture in Functional Mathematics―　23pages	April 30, 2009
COE Lecture Note Vol.20	Yasuhide Fukumoto Yasunori Maekawa	Math-for-Industry Tutorial: Spectral theories of non-Hermitian operators and their application　184pages	June 19, 2009
COE Lecture Note Vol.21	Faculty of Mathematics, Kyushu University	Forum “Math-for-Industry” Casimir Force, Casimir Operators and the Riemann Hypothesis　95pages	November 9, 2009