LNCS 3718

Victor G. Ganzha
Ernst W. Mayr
Evgenii V. Vorozhtsov (Eds.)

Computer Algebra in Scientific Computing

8th International Workshop, CASC 2005
Kalamata, Greece, September 2005
Proceedings

Springer

Victor G. Ganzha Ernst W. Mayr
Evgenii V. Vorozhtsov (Eds.)

Computer Algebra in Scientific Computing

8th International Workshop, CASC 2005
Kalamata, Greece, September 12-16, 2005
Proceedings

Springer

Volume Editors

Victor G. Ganzha
Ernst W. Mayr
Technische Universität München
Institut für Informatik
Garching, Germany
E-mail:{ganzha, mayr}@in.tum.de

Evgenii V. Vorozhtsov
Russian Academy of Sciences
Institute of Theoretical and Applied Mechanics
Novosibirsk, Russia
E-mail: vorozh@itam.nsc.ru

Library of Congress Control Number: 2005932112

CR Subject Classification (1998): I.1, F.2.1-2, G.1, I.3.5, I.2

ISSN	0302-9743
ISBN-10	3-540-28966-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-28966-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11555964 06/3142 5 4 3 2 1 0

Preface

CASC 2005 continued a tradition — started in 1998 — of international conferences on the latest advances in the application of computer algebra systems (CASs) and methods to the solution of various problems in scientific computing.

The methods of scientific computing play an important role in research and engineering applications in the natural and the engineering sciences. The significance and impact of computer algebra methods and computer algebra systems for scientific computing has increased considerably in recent times. Nowadays, such general-purpose computer algebra systems as Maple, Magma, Mathematica, MuPAD, Singular, CoCoA and others enable their users to solve the following three important tasks within a uniform framework:

(a) symbolic manipulation;
(b) numerical computation;
(c) visualization.

The ongoing development of such systems, including their integration and adaptation to modern software environments, puts them at the forefront in scientific computing and enables the practical solution of many complex applied problems in the domains of natural sciences and engineering.

Greece offers excellent infrastructures for hosting international conferences, and this was a reason for us to choose the city of Kalamata, Greece, as the location for CASC 2005, the eighth conference in the sequence of CASC conferences. The seven earlier CASC conferences, CASC 1998, CASC 1999, CASC 2000, CASC 2001, CASC 2002, CASC 2003, and CASC 2004 were held, respectively, in St. Petersburg, Russia, in Munich, Germany, in Samarkand, Uzbekistan, in Konstanz, Germany, in the Crimea (Ukraine), in Passau (Germany), and in St. Petersburg, Russia, and they proved to be successful.

The Program Committee did a tremendous job reading and evaluating 75 submitted papers, as well as soliciting external reviews, and all of this in a very short period of time. There were about three reviews per submission on average. The result of this job is reflected in this volume, which contains revised versions of the accepted papers. The collection of papers included in the proceedings covers various topics of computer algebra methods, algorithms, and software applied to scientific computing:

1. algebraic methods for nonlinear polynomial equations and inequalities;
2. symbolic-numeric methods for differential and differential-algebraic equations;
3. algorithmic and complexity considerations in computer algebra;
4. algebraic methods in geometric modelling;
5. aspects of computer algebra programming languages;
6. automatic reasoning in algebra and geometry;

7. complexity of algebraic problems;
8. exact and approximate computation;
9. parallel symbolic-numeric computation;
10. Internet accessible symbolic and numeric computation;
11. problem-solving environments;
12. symbolic and numerical computation in systems engineering and modelling;
13. computer algebra in industry;
14. solving problems in the natural sciences;
15. numerical simulation using computer algebra systems; and
16. mathematical communication.

This workshop, like the earlier CASC workshops, was intended to provide a forum for researchers and engineers in the fields of mathematics, computer science, numerical analysis, and industry, to interact and exchange ideas. An important goal of the workshop was to bring together all these specialists for the purpose of fostering progress on current questions and problems in advanced scientific computing.

CASC 2005 featured two satellite workshops

- *Algebraic and Matrix Computation with Applications*, organized by I.Z. Emiris, B. Mourrain, and M.N. Vrahatis

- *Kalamata Combinatorics*, organized by I.S. Kotsireas and C. Koukouvinos

Researchers from France, Germany, Italy, Greece, Spain, Russia, Japan, USA, Canada, Czech Republic, and Egypt participated in CASC 2005.

CASC 2005 wishes to acknowledge generous support from sponsors:

- Hellenic Ministry of Culture, *ΥΠΠΟ*, Athens, Greece
- Maplesoft, Waterloo, Ontario, Canada
- National and Kapodistrian University of Athens, Greece
- University of Patras, Greece
- Wilfrid Laurier University, Waterloo, Ontario, Canada

Our particular thanks are due to the CASC 2005 conference chairs and members of the Local Organizing Committee I.Z. Emiris (Athens), I.S. Kotsireas (Waterloo), and M.N. Vrahatis (Patras), who ably handled local arrangements in Kalamata. We also thank the members of the General Organizing Committee, W. Meixner and A. Schmidt, in particular for their work in preparing the conference proceedings.

Munich, July 2005

V.G. Ganzha
E.W. Mayr
E.V. Vorozhtsov

Organization

CASC 2005 was organized jointly by the Department of Informatics at the Technische Universität München, Germany, and the Computer Algebra Research Group of Wilfried Launer University, Canada.

Workshop General Chairs

Vladimir Gerdt (JINR, Dubna)
Ernst W. Mayr (TU München)

Program Committee

Gerd Baumann (Cairo)
Laurent Bernardin (Waterloo)
Hans-Joachim Bungartz (Munich)
Andreas Dolzmann (Passau)
Victor Edneral (Moscow)
Ioannis Emiris (Athens)
Jean-Charles Faugere (Paris)
Victor Ganzha (Munich, co-chair)
Laureano Gonzalez-Vega (Santander)
Evgenii Grebenikov (Moscow)
Jaime Gutierrez (Santander)
Ilias Kotsireas (Waterloo)
Robert Kragler (Weingarten)
Richard Liska (Prague)
Bernard Mourrain (Sophia Antipolis)
Patrick O'Leary (Galway)
Eugenio Roanes-Lozano (Madrid)
Yosuke Sato (Tokyo)
Werner Seiler (Heidelberg)
Stanly Steinberg (Albuquerque)
Nikolay Vassiliev (St. Petersburg)
Gilles Villard (Lyon)
Evgenii Vorozhtsov (Novosibirsk, co-chair)
Michael N. Vrahatis (Patras)
Andreas Weber (Bonn)

Conference Chairs and Local Organizing Committee

I.Z. Emiris (Athens)
M.N. Vrahatis (Patras)
I.S. Kotsireas (Waterloo)

General Organizing Committee

Werner Meixner (Munich, chair)
Annelies Schmidt (Munich, secretary)

Electronic Address

WWW site: `http://wwwmayr.in.tum.de/CASC2005`

Table of Contents

On Regular and Logarithmic Solutions of Ordinary Linear Differential Systems*

S.A. Abramov[1], M. Bronstein[2], and D.E. Khmelnov[1]

[1] Dorodnicyn Comp. Center of the Russ. Acad. of Sciences, Moscow 119991, Russia
{sabramov, khmelnov}@ccas.ru
[2] INRIA – Café, BP 93, 06902-Sophia Antipolis Cedex, France

Abstract. We present an approach to construct all the regular solutions of systems of linear ordinary differential equations using the desingularization algorithm of Abramov & Bronstein (2001) as an auxiliary tool. A similar approach to find all the solutions with entries in $C(z)[\log z]$ is presented as well, together with a new hybrid method for constructing the denominator of rational and logarithmic solutions.

1 Introduction

Let C be an algebraically closed field of characteristic 0, z be an indeterminate over C, and

$$L = Q_\rho(z)D^\rho + \cdots + Q_1(z)D + Q_0(z), \tag{1}$$

where $D = d/dz$ and $Q_\rho(z), \ldots, Q_0(z) \in C[z]$. A *regular solution* of $Ly = 0$ (or of L) at a given point $z_0 \in C$, is a solution of the form $(z - z_0)^\lambda F(z)$ with $F(z) \in C((z - z_0))[\log(z - z_0)]$, where $C((z - z_0))$ is the field of (formal) Laurent series over C. If $F(z)$ has valuation 0, then λ is called the *exponent* of the regular solution (otherwise it is an exponent modulo $\mathbb{Z}$). Using the change of variable $\bar{z} = z - z_0$, we can assume without loss of generality that $z_0 = 0$. The problem of constructing all the regular solutions is solved by the Frobenius algorithm (1873, [8, Chap.IV],[9],[14, Chap.V]), which is based on the indicial equation $f(\lambda) = 0$ of L at 0. Not only the roots of $f(\lambda) = 0$, each taken separately, are substantial for the Frobenius algorithm, but also their multiplicities and whether some roots differ by integers. Later, in 1894, L. Heffter proposed another algorithm to solve the same problem ([10, Kap.II,VIII],[14, Chap.V]). For a given root λ of the indicial equation, Heffter's algorithm constructs a basis (possibly empty) for all the regular solutions with exponent λ. Once λ is fixed, that algorithm does not depend on the multiplicity of λ, nor on the existence of another root at an integer distance from λ. It constructs a sequence $E_0, E_1, \ldots$ of linear differential equations, whose right-hand side contains solutions of the preceding equations. If

$$z^\lambda \left(g_0(z) + g_1(z)\frac{\log z}{1!} + g_2(z)\frac{\log^2 z}{2!} + \cdots + g_m(z)\frac{\log^m z}{m!} \right)$$

* Work partially supported by the ECO-NET program of the French Foreign Affairs Ministry, project No. 08119TG, and by RFBR grant No. 04-01-00757.

is a regular solution of (1) then $g_i(z) \in C((z))$ is a solution of E_i for each i. All the regular solutions of (1) with exponent λ have been found when we reach an equation E_{m+1} that has no nonzero Laurent series solution.

To apply the original Frobenius or Heffter algorithm at an arbitrary singularity of a system of linear differential equations would require transforming the system to a scalar differential equation (*e.g.* by the cyclic vector method). That scalar equation usually has huge coefficients, making this approach quite unpractical. If $z = 0$ is a regular singularity of a first-order system of the form

$$\frac{dY}{dz} = A(z)Y(z), \quad A(z) \in \mathrm{Mat}_N(C(z)) \tag{2}$$

then, it is possible in theory to use a variant of the Frobenius algorithm that can be applied directly [8, p. 136, exercise 13], but this approach cannot be applied to irregular singularities or higher-order systems.

A generalization of Heffter's algorithm for constructing the regular solutions of first order systems of the form (2) is described in [5, §5] (an extended version is in [6, §3]). That algorithm is direct, *i.e.* it does not use any uncoupling procedure. A necessary step is however to find all the Laurent series solutions of a given system. For this task and for producing the indicial equation $f(\lambda) = 0$, the algorithm of [6,5] transforms the system into its *super-irreducible* form (see [11]).

We describe in this paper another adaptation of Heffter's algorithm to linear differential system, which uses the desingularization algorithm of [2,3] instead of transforming a system into its super-irreducible form. This allows us to handle higher order systems, *i.e.* operators such as (1) where the Q_i are matrices of polynomials, directly, *i.e.* without converting them to larger first-order systems. We study the efficiency of the approach both from the theoretical and practical viewpoints, and it shows that solving them directly is more efficient.

In a similar way, we solve the related problem of finding all the solutions with entries in $C(z)[\log z]$, which we call *logarithmic* solutions. This problem is decomposed into first finding a universal denominator for the solutions, and then finding solutions with entries in $C[z][\log z]$. The latter problem is solved by a slightly modified version of our algorithm for the regular solutions. For the denominator, in addtion, we propose a new hybrid method that combines the algorithm of [2] with a reduction algorithm specific to regular singularities [7]. The hybrid method is applicable to the case of first order systems, and in this case it speeds up the computation quite often (see Sect. 5).

2 Desingularization of Linear Recurrence Systems

Linear recurrences with variable coefficients are of interest for many applications (*e.g.* combinatorics and numeric computation). Consider a recurrence of the form

$$P_l(n)x_{n+l} + P_{l-1}(n)x_{n+l-1} + \cdots + P_t(n)x_{n+t} = r_n \tag{3}$$

where $l \geq t$ are arbitrary integers, $x = (x^1, \ldots, x^N)^T$ is a column vector of unknown sequences (such that $x_i = (x_i^1, \ldots, x_i^N)^T$), $P_t(n), \ldots, P_l(n) \in \mathrm{Mat}_N(C[n])$,

$P_t(n) \neq 0 \neq P_l(n)$ and $r_n \in C[n]^N$. The matrices $P_l(n)$ and $P_t(n)$ are called respectively the *leading* and *trailing* matrices of the recurrence. When $P_t(n)$ and $P_l(n)$ are nonsingular, the roots of their determinants are important for determining the structure of the solution space, as they give bounds on the solutions whose support is bounded above or below. It may happen however that $P_t(n)$ or $P_l(n)$ is singular (or both). In that case, they do not yield bounds on the solutions, but it is also difficult, from a computational standpoint, to use the recurrence (3) to compute the sequence of vectors that it generates. A natural solution in that case is to transform the recurrence system into an equivalent one with either the leading or trailing matrix nonsingular. That transformation may be a "quasi–equivalence", in the sense that the eventual changes in the solution space can be easily described. Such a transformation (the EG-algorithm) was developed in [1] and later improved in [2]. In addition to the transformed system, it also yields a finite set of linear constraints such that the solutions of the original system are exactly those of the transformed system that also satisfy the new constraints (each of the constraints is a linear relation that contains a finite set of variables x_i^j).

3 Regular Solutions

We consider in this section the higher order system $LY = 0$ where L is of the form (1) with $Q_0, \ldots, Q_\rho \in \mathrm{Mat}_N(C[z])$ and Q_ρ nonsingular.

3.1 Description of the Algorithm

Using the standard basis $(z^m)_{m \geq 0}$ of $C[z]$, we construct (see [2, §2]) its associated recurrence system $Rc = 0$, where $R = P_l(n)E^l + \cdots + P_t(n)E^t$, E is the shift operator and $P_j(n) \in \mathrm{Mat}_N(C[n])$ for $t \leq j \leq l$. If $\det P_l(n)$ is identically 0, then it is possible (see Section 2) to transform the recurrence system into an equivalent one (together with a finite set of linear constraints) with $\det P_l(n) \neq 0$, so assume from now on that $\varphi(n) = \det P_l(n) \neq 0$. Let $\psi(n) = \varphi(n - l)$ and n_0, n_1 be respectively the minimal and maximal integer roots of $\psi(n)$ (if there is no integer root, then $LY = 0$ has no Laurent series solution). Any Laurent series solution of $LY = 0$ has no term $c_k z^k$ with $c_k \in C^N$ and $k < n_0$. Using the recurrence $Rc = 0$ and the additional constraints, we can, by a linear algebra procedure, compute a basis of the linear space of initial segments $c_{n_0} z^{n_0} + c_{n_0+1} z^{n_0+1} + \cdots + c_M z^M$, where M is a fixed integer, chosen greater that n_1 and all the indices appearing in the linear constraints. Observe that if our differential system is inhomogeneous with a Laurent series right-hand side (whose coefficients are given by a linear recurrence system), then we can similarly construct a basis of the affine space of its Laurent series solutions. If $\psi(n)$ has a non-integer root λ, then the preliminary change of variable $Y = z^\lambda \bar{Y}$ produces a new system for $\bar{Y}$, hence a new recurrence with a new $\bar{\psi}(n) = \psi(n - \lambda)$. Therefore, we can always work with the integer roots of ψ. For any integer $m \geq 0$, the result of applying L to $g(z) \log^m(z)/m!$ is clearly of the form

$$L_{m,m}(g) \frac{\log^m z}{m!} + \cdots + L_{m,1}(g) \frac{\log z}{1!} + L_{m,0}(g), \tag{4}$$

where the coefficients of the differential operators $L_{i,j}$ belong to $\mathrm{Mat}_N(C(z))$. Proofs of the following proposition can be found in [10] and [12, Sect. 3.2.1].

Proposition 1. *The coefficients of all the $L_{i,j}$ in (4) belong to* $\mathrm{Mat}_N(C[z, z^{-1}])$. *In addition, $L_{0,0} = L$ and $L_{i+j,j} = L_{i,0}$ for any $i, j \geq 0$.*

Let $L_i = L_{i,0} (= L_{i+j,j}$ for any $j \geq 0)$. Using (4) and Proposition 1 we obtain

$$L\left(\sum_{m=0}^{k} g_{k-m}(z)\frac{\log^m z}{m!}\right) = \sum_{m=0}^{k}\left(\sum_{j=0}^{k-m} L_j(g_{k-m-j})\right)\frac{\log^m z}{m!}\,.$$

Therefore,

$$Y = \sum_{m=0}^{k} g_{k-m}(z)\frac{\log^m z}{m!} \tag{5}$$

is a solution of $LY = 0$ if and only if $(g_0(z), \ldots, g_k(z))$ is a Laurent series solution of the inhomogeneous linear system

$$L_0(g_i) = -\sum_{j=1}^{i} L_j(g_{i-j}) \quad \text{for } 0 \leq i \leq k\,. \tag{6}$$

When we find $g_0(z)$ using the first equation $L_0(g_0) = 0$ of (6), that solution contains arbitrary constants. When we use $g_0(z)$ in the right-hand side of the next equation $L_0(g_1) = -L_1(g_0)$ of (6) those arbitrary constants appear linearly in the right-hand side. Using the same technique as when solving such scalar parametric inhomogeneous equations (see for example [4]), we find together with $g_1(z)$ linear constraints on the arbitrary constants appearing in g_0 and g_1. Repeating this process, we find at each step that $g_0, \ldots, g_i$ depend on unknown constants together with a linear system for those constants. In order for this process to terminate, we need to ensure that we always reach an integer k such that (6) has no Laurent series solution with $g_0 \neq 0$. Heffter proved this in the scalar case, and his proof carries over to systems.

Proposition 2. *The set $K = \{k \geq 0$ such that (6) has a solution with $g_0 \neq 0\}$ is finite. If K is empty, then $LY = 0$ has no nonzero solution in $C((z))[\log z]$. Otherwise, $K = \{0, \ldots, \mu\}$ for some $\mu \geq 0$ and any solution in $C((z))[\log z]$ of $LY = 0$ has the form (5) where $(g_0, \ldots, g_\mu)$ is a solution of (6) with $k = \mu$. In addition, any solution of (6) with entries in $C((z))$ generates a solution of $LY = 0$.*

Proof. Let G_k be the linear space of all the (regular) solutions of the form (5) of $LY = 0$, and $Y \in G_k$. Writing $Y = \sum_{m=0}^{k+1} h_{k+1-m}(z)\log^m(z)/m!$ where $h_0 = 0$ and $h_{i+1} = g_i$ for $0 \leq i \leq k$, we see that $G_0 \subseteq G_1 \subseteq \cdots \subseteq G_k \subseteq G_{k+1} \subseteq \cdots$. Let $k > 0$ be in K and $(g_0, \ldots, g_k)$ be a solution of (6) with $g_0 \neq 0$. Then, $(g_0, \ldots, g_{k-1})$ is a solution of (6) with $k - 1$ and so on, which implies that $\{0, \ldots, k\} \subset K$ and that $\dim_C G_k \geq k$. This produces k linearly independent solutions of the form (5) of $LY = 0$. On the other hand, since $LY = 0$ is equivalent to a scalar equation of order at most ρN, $\dim_C G_k \leq \rho N$ for all k, so K is either empty or of the form $\{0, \ldots, \mu\}$ for some $\mu \leq \rho N$. Therefore, if we compute $G_0, G_1, \ldots$ using (6), we eventually find an integer $k \leq \rho N$ for which the system (6) has a solution only for $g_0 = 0$. At this point, G_{k-1} contains all the solutions of $LY = 0$ with entries in $C((z))[\log z]$.

Summarizing, our scheme for constructing regular solutions of $LY = 0$ is:

1. Construct its associated matrix recurrence in the form (3) and transform it into an equivalent one with nonsingular leading matrix (see Section 2). Let

$$P'_l(n) z_{n+l} + P'_{l-1}(n) z_{n+l-1} + \cdots + P'_t(n) z_{n+t} = r'_n \tag{7}$$

   be the resulting recurrence (it may include in addition a set of linear constraints). Compute all roots of $\varphi(n) = \det(P'_l(n))$, divide them into groups having integer differences, and construct the set Λ consisting of one representative for each groups.
2. For each $\lambda \in \Lambda$, compute regular solution whose exponent is λ:
   (a) Compute a system S_λ by substituting $Y = z^\lambda Y_\lambda$ and by following multiplication of the system by $z^{\rho-\lambda}$. This induces a transformation of (7). We get a recurrence R_λ: $P''_l(n) z_{n+l} + P''_{l-1}(n) z_{n+l-1} + \cdots + P''_t(n) z_{n+t} = r'_n$, where $P''_i(n) = P'_i(n + \lambda)$, $i = l, l-1, \ldots, t$, and since $P'_l(n)$ is nonsingular, so is $P''_l(n)$ and no additional desingularisation is required. The transformed recurrence R_λ may include a set of linear constraints which are transformed correspondingly.
   (b) Determine the number M_λ of required initial terms of Laurent series, which is greater than all the integer roots of the determinant of the leading matrix of R_λ as well as all the indices appearing in the additional constraints.
   (c) Successively solve systems (6) for the required number of terms of its Laurent series solutions, using the recurrence R_λ while it is possible. This yields regular solutions y_λ of S_λ in the form (5).
3. Combine all the solutions into the general regular solution $y = \sum_{\lambda \in \Lambda} z^\lambda y_\lambda$.

Note that we construct regular solutions by representing all involved Laurent series by truncated expansions until an appropriate terms, such that the number of linearly independent solutions is determined correctly and computation of the subsequent terms can be performed one by one by means of recurrenses $\{R_\lambda\}$.

3.2 Computing and Implementation Remarks

The Associated Recurrence System. The main part of the algorithm is solving the individual systems from the sequence (6). Note however that all those systems have in the left-hand side the same operator $L_0 = L$. Therefore, the associated linear recurrence systems also have the same left-hand sides, but their right-hand sides are different. In order to desingularize the recurrence system only once, we apply during the first desingularization (step 1 above) all the transformations to a generic right-hand side. As a result, we have the transformed recurrence with a non-singular leading matrix, a finite set of linear constraints and the transformed right-hand side in a generic form. Each component of this generic transformed right-hand side is a linear combination of possibly shifted components of the original right-hand side before desingularization. In this way we can use the same transformed recurrence for solving any system from the sequence (6) specifying the concrete right-hand side by substituting the corresponding values into the generic right-hand side (such substitutions commute with any elementary transformation that our desingularization algorithm uses).

Computing the Right-Hand Sides. This is not so simple since the right-hand side for the i-th system in (6) is $h_i = -\sum_{j=1}^{i} L_j(g_{i-j})$, where $g_0, \ldots, g_{i-1}$ are Laurent series solutions of the preceding systems. Since we represent (truncated) Laurent series solution by segment of initial terms, we need to determine the required numbers of initial terms of $g_0, \ldots, g_{i-1}$. That number is determined in an algorithmic way and depends on the number M_λ of initial terms of the transformed right-hand side, which is determined in step 2b of the algorithm for all the systems in the sequence (6), and ensures that the next terms of the series are computed from the preceding ones by a simple use of the recurrence. So we compute the transformed right-hand side in the following way:

1. Taking into account M_λ and the components of the transformed generic right-hand side, compute the numbers of required initial terms of the components of the right-hand side before transformation, to ensure that the number of initial terms in the transformed right-hand side is equal to M_λ.
2. Taking into account the form of the operators $L_1, \ldots, L_i$, compute the numbers of initial terms of $g_0, \ldots, g_{i-1}$ required to ensure the needed numbers of initial terms of the components of right-hand side before transformation.
3. Compute the corresponding initial segments of $g_0, \ldots, g_{i-1}$.
4. Compute the initial segment of the right-hand side before transformation substituting the initial segments of $g_0, \ldots, g_{i-1}$ into h_i.
5. Compute the initial segment of the transformed right-hand side substituting the initial segment of the right-hand side before transformation into the transformed generic right-hand side.

Extending Solution Components. Computing the transformed right-hand side depends on computing the initial segments of $g_0, \ldots, g_{i-1}$ (step 3 in Section 3.2). Since the required number of initial terms of the solution component g_k may be greater than the number of the initial terms computed in the preceding steps of the algorithm, we need to extend that component, which requires computing additional terms using the associated recurrence. This means in turn that we need to extend the corresponding transformed right-hand side, computing it using the approach from Section 3.2 while replacing M_λ by the new number. Note that this may again require extending other solution components, so this procedure is recursive.

Computing Initial Segments. When we compute the transformed right-hand side of the recurrence, solving a system from the sequence (6) for the required number of initial terms can be done step by step using the recurrence. In each step one of the following options occurs:
- the next term is computed as a function of the previous terms;
- a linear constraint on the previous terms appears, which can be either solved or inconsistent, in which case there is no Laurent series solution;
- the next term is a new arbitrary constant (which may be specified later in the computation when solving constraints).
After all those steps, either all the initial terms have been computed (some of them being arbitrary constants) or we have proven that there is no Laurent series solution. Note that:

1. Since each of the $g_0, \dots, g_{i-1}$ may have arbitrary constants, the right-hand side for the i-th system in the sequence (6) may have the same arbitrary constants. This leads to the fact that, during the computation of the Laurent series solution of the i-th system, some of the arbitrary constants may be specified when resolving newly appearing constraints. This could turn the current initial segment of g_0 to zero. Since the number of terms in the segment is such that all the remaining terms are computed by the associated recurrence whose leading matrix is non-singular, this implies that g_0 is identically zero. As noted earlier, when this happens, then all the solution components have been computed and i-th system has no Laurent series solutions with $g_0 \neq 0$.
2. As noted earlier, since $G_k \subset G_{k+1}$, we can use only the last solution found of the form (5) with $g_0 \neq 0$ as the regular solution whose exponent is λ, since it also contains all the previously found solutions of that form.

4 Logarithmic Solutions

Finding regular solutions is a local problem. We consider in this section the analogous global problem of finding solutions whose entries are in $C(z)[\log z]$ of the higher order system $LY = 0$ where L is of the form (1) with $Q_0, \dots, Q_\rho \in \mathrm{Mat}_N(C[z])$ and Q_ρ nonsingular. In the case of first-order systems $Y' = AY$, an algorithm, based on super-irreducible forms, for computing such solutions was presented in [5]. As for regular solutions, we present here an alternative that does not require conversion to first order systems, and that uses desingularization (see Section 2) instead of super-irreducible forms. Recall that finding rational solutions of $LY = 0$ proceeds in two distinct steps: we construct first a universal denominator, *i.e.* $d \in C[z]$ such that $dY \in C[z]^N$ for any solution $Y \in C(z)^N$ of $LY = 0$. We then search for the polynomial solutions of the system obtained by the change of variable $\bar{Y} = dY$ in L. As remarked in [5, §5.1], a universal denominator for rational solutions is also valid for logarithmic solutions. As described in [2, §8.1], the irreducible factors of a universal denominator must all divide $\det(Q_\rho)$ and we can compute such a universal denominator using the algorithm described there. This reduces our problem to finding solutions whose entries are in $C[z][\log z]$. Before solving that problem in Section 4.2, we first present a heuristic for accelerating the computation of universal denominators in the case of first-order systems.

4.1 A Hybrid Heuristic for First-Order Systems

We restrict in this section our systems to be of the form (2). If all the finite singularities of that system are known to be regular, then the reduction method of [7] transforms all of them *simultaneously* into simple poles, that is, it returns a change of variable that transforms the system into an equivalent one where the denominator of A is squarefree. In that case, the exponents at all the singularities can be easily computed as eigenvalues of the associated residue matrices, yielding a fast way to compute a universal denominator. That algorithm repeats the following *single reduction step*: as long as the denominator of A is not squarefree, choose a row of A whose common denominator $d \in C[z]$ is not squarefree. Using only extended gcd computations, compute an invertible $T \in \mathrm{Mat}_N(C(z))$ whose rows form a $C[z]$-basis of the free $C[z]$-module $C[z]^{1\times N} + C[z]\omega$ where $C[z]^{1\times N}$ is the module of row-vectors and ω is the selected row of A multiplied by the squarefree part of d. Apply then the change of variable $\bar{Y} = TY$ to $dY/dz = AY$ and repeat this process, which is shown in [7] to yield a matrix with a squarefree denominator after finitely many steps.

Not much is known about the behavior of that algorithm in the presence of irregular singularities (except that it cannot terminate!) but we use it here as a heuristic to separate the (proven) regular and (putative) irregular singularities of the system. Since the proven regular singularities are transformed into simple poles in this process, we use the eigenvalue approach to compute their exponents, limiting the use of desingularization to the remaining ones. Note that wrongly classifying a singularity as irregular does not affect the correctness of the hybrid method, since going through the recurrence approach at such singularities yield a correct bound. This yields the following hybrid heuristic for the universal denominator, given a first-order system in the form (2):

1. Initialize the set of proven regular singularities to be $R := \emptyset$ and the universal denominator to be $U := 1$.
2. Compute an irreducible factorisation of the denominator of A, the set F of its irreducible factors and the subset F_1 of factors having multiplicity 1.
3. Initialize the set $S := F$ of "unclassified" singularities.
4. Repeat the following steps:
   (a) For each factor in $f \in F_1$ (proven regular singularities) compute its exponent $e \geq 0$ in the universal denominator by the classical approach for simple poles. Update $U := Uf^e$.
   (b) Update $R := R \cup F_1$ followed by $S := S \backslash R$.
   (c) Repeat the single reduction step of [7] described above, limiting it to the factors in S (see below), and get the transformed system with matrix A_r until either (i) a new squarefree factor of the denominator of A_r is found, or (ii) we have determined that all the elements of S correspond to putative irregular singularities, using the following heuristic: for each $f \in S$ compute

  - mf — minimal positive multiplicity of f in the denominators of the rows of the original A
  - mf' — the same but for the rows of the transformed A_r
  - sf — sum of the multiplicities of f in the denominators of the rows of the original A
  - sf' — the same but for the rows of the transformed A_r

  We declare f to be a putative irregular singularity if one of the following conditions hold:
  - $mf' > mf$;
  - $mf' = mf$ and $sf' > sf$;
  - $mf' = mf$, $sf' = sf$ and this situation is repeated for the third successive single reduction step (note that we count the number of such successive stabilities of mf and sf and pass that count from one reduction step to the next one)

  If f has been declared to be a putative irregular singularity, then we remove it from S for the next single reduction step.

  (d) Let $F_1 \subset F\backslash R$ be the set of factors of the denominator of A_r with multiplicity 1. If $F_1 \neq \emptyset$, then go back to step 4a otherwise leave the loop.

5. For each $f \in F\backslash R$ (putative irregular singularities) compute its exponent $e \geq 0$ in the universal denominator by the desingularization approach of [2] (see Sect. 2). Update $U := Uf^e$.

Note that the above algorithm is computing the exponent of factors reduced to simple poles as soon as they appear during the reduction, so we can exclude them from the following reductions. This is more efficient than computing them at the end since each single reduction step increases the degrees and coefficients of the entries of A_r. We can use the exponents computed using the transformed matrix A_r at each step as the correct exponents in a universal denominator for the original matrix A because the change of variables at each single step are given by inverses of polynomial matrices [7, Theorem 2].

To limit the single reduction step to the factors in $S \subset F$ at step 4c, we adapt the single reduction step as follows: the denominator d of each row of A can be factored as $d = \prod_{p \in S} p^{e_p} \prod_{q \notin S} q^{e_q}$. Choose a row of A for which $e_p > 1$ for some $p \in S$, let ω be that row of A multiplied by $\prod_{p \in S} p^{\min(e_p,1)} \prod_{q \notin S} q^{e_q}$ and compute a basis of $C[z]^{1 \times N} + C[z]\omega$ as explained above.

A final remark about the hybrid method: there is some arbitrariness in selecting the row used in the single reduction step. In our implementation, we select the row with the maximal sum of the multiplicities of the factors from S in its denominator.

4.2 Solutions with Entries in C[z][log z]

Since our algorithm for finding regular solutions at $z = 0$ returns truncated Laurent series, it can easily be adapted to return only the solutions with entries in $C[z][\log(z)]$. Only the following changes are needed:

- Instead of dynamically bounding the number of terms of the series to compute, we compute an upper bound of the degrees of the polynomial solutions and use that number of terms. Such an upper bound is computed from the roots of the determinant of the *trailing* matrix of the associated recurrence system (see [1,2] for details).
- As we are not interested in the z^λ factors, we skip the computation of the set Λ of exponents modulo $\mathbb{Z}$ at $z = 0$.

This yields the following algorithm for solutions in $C[z][\log z]$:

1. Construct its associated matrix recurrence in the form (3) and transform it into an equivalent one R with $P_t(n)$ nonsingular (see Section 2).
2. Compute an upper bound M on the degree of the polynomial solutions from the integer roots of $\det P_t(n)$.
3. Successively solve systems (6) for their polynomial solutions, using the recurrence R while it is possible. This means setting all the coefficients of the series with negative indices to 0 and computing truncated series up to the degree bound. As we compute successive right-hand sides, we can refine our degree bound. Since the polynomials are computed completely, there is no need to extend them as we compute additional right-hand sides. As we used a recurrence with nonsingular trailing matrix to bound the degree, it is natural to use it again in this step to compute the coefficients of the polynomials. So we compute them starting with the coefficients of highest degree and work down to the constant coefficients. For regular solutions, we used a nonsingular leading matrix to get the exponents and precisions, so we obtained the coefficients in the reverse order.

5 Complexity and Experimental Comparisons

Consider the higher order system $LY = 0$ where L is of the form (1) with $Q_0, \dots, Q_\rho \in \mathrm{Mat}_N(C[z])$ and Q_ρ nonsingular. Let δ be a bound on the degrees of the entries of the Q_i's. There is no known complete complexity analysis for our method, nor for the method of [6]. However, since our method computes only one desingularisation (see Sect. 3.1) and the method of [6] computes only one super-irreducible form, we attempt here to compare those operations.

The basic operation for desingularisation is computing ranks and nullspaces of matrices of polynomials. Using [15], this has a complexity of $\mathcal{O}^\sim(n^\omega d)$ operations in C, where n is the size of the matrix, d a bound on the degree of its entries, ω the exponent of matrix multiplication and the $\mathcal{O}^\sim$ notation means that we ignore logarithmic factors. The leading matrix of our recurrence is of size N and its entries are of degree bounded by ρ (and not δ since the transformation from differential equations to recurrences sends D to n), so the first nullspace has a cost of $\mathcal{O}^\sim(N^\omega \rho)$.

The basic operation for super-irreducible forms [11] is computing characteristic polynomials of matrices of polynomials. Using [13] this has a a complexity of $\mathcal{O}^\sim(n^{\omega+1/3} d)$ operations in C. Since $LY = 0$ must first be converted to a first order system of size $N\rho$, the first characteristic polynomial has a cost of $\mathcal{O}^{\sim}(N^{\omega+1/3}\rho^{\omega+1/3}\delta)$.

In practice, we expect desingularization and super-irreducible forms to converge quickly, *i.e.* to compute relatively few nullspaces or characteristic polynomials. In theory, desingularization can require $N(\delta+\rho)$ nullspaces in the worst case. Since we do not know the growth of the degrees of the entries, this only yields the approximate complexity of $\mathcal{O}^{\sim}(N^{\omega+1}(\delta+\rho)F(\rho))$ where $F(\rho)$ encodes the growth in the degrees. Similarly, super-irreducible forms can require $s(s+1)/2$ characteristic polynomials in the worst case, where s is the polar order of the matrix at $z=0$, which can be as high as δ. This then yields an approximate complexity of $\mathcal{O}^{\sim}(N^{\omega+1/3}\rho^{\omega+1/3}G(\delta)^3)$ where $G(\delta)$ encodes the growth in the degrees.

We have implemented our regular solution algorithm in Maple on top of the package `LinearFunctionalSystems`, which contains solvers based on [2]. Our function returns the regular solutions with truncated Laurent series of a linear differential system with polynomial coefficients. The number of terms of the series is determined automatically to ensure that the remaining terms of the series can be computed using the associated recurrences (*i.e.* its leading matrix is invertible for all the remaining terms). In order to extend initial segments of the Laurent series, another function is provided, which returns the regular solution with the series extended to a given degree.

For comparison purposes, we used the Maple package `ISOLDE`, which implements the algorithm of [6]. We compared the two programs on several sets of generated systems [1]. Although our program is faster in the majority of examples, the gains for first-order systems are only by a small constant factor. Moreover, since the programs use different approaches, this comparison can only conclude that the two methods are of comparable efficiency for first-order systems and their weak and strong features are displayed on different systems. For exmaple, for one of the sets, most systems were solved faster by our program, but `ISOLDE` had a lower total CPU time for one of the subsets since it solved much faster a few systems in that subset. Those results indicate as well the difficulty of developping a polyalgorithm that would automatically detect the most efficient method to use for each particular input. The main advantage of our method is however its direct applicability to higher-order systems, where the experimental results confirm the better efficiency. For the comparison we generated a set of higher-order systems and as well transformed all the systems in the set to corresponding first order systems. Then we solved the higher-order systems directly by our program and solved the corresponding first-order systems both by our program and by `ISOLDE`. All systems in the set were solved faster by direct approach. To be fair, we should remark that our program has been steadily improved by the recent update of some modules, while `ISOLDE` has not been updated for a long time. So we do not exclude the possibility that further improvements in `ISOLDE` could lead to some changes in our comparisons. They nevertheless reflect accurately the

[1] All comparisons are available at `www.ccas.ru/sabramov/casc2005.html`

current status of those programs, as well as parallel similar conclusions obtained by comparing them on the computation of rational solutions [3].

We have also implemented our logarithmic solution algorithm on top of the package `LinearFunctionalSystems`, together with an option in the corresponding procedure `UniversalDenominator` to allow the use of the hybrid heuristic. Again we generated several sets of systems, and solved them with and without the hybrid heuristic for the universal denominator. The results showed that the hybrid method generally yields a significant speedup, though it might lead to additional work without any gain sometimes.

References

1. Abramov, S.: EG–eliminations. Journal of Difference Equations and Applications **5** (1999) 393–433
2. Abramov, S., Bronstein, M.: On solutions of linear functional systems. In Proceedings of ISSAC'2001, ACM Press (2001) 1–6
3. Abramov, S., Bronstein, M., Khmelnov, D.: Regularization of linear recurrence systems. In Transactions of the A.M. Liapunov Institute. Volume 4. (2003) 158–171
4. Abramov, S., Bronstein, M., Petkovšek, M.: On polynomial solutions of linear operator equations. In Proceedings of ISSAC'95, ACM Press (1995) 290–296
5. Barkatou, M.A.: On rational solutions of systems of linear differential equations. Journal of Symbolic Computation **28** (1999) 547–567
6. Barkatou, M., Pflügel, E.: An algorithm computing the regular formal solutions of a system of linear differential equations. Journal of Symbolic Computation **28** (1999) 569–587
7. Bronstein, M., Trager, B.: A reduction for regular differential systems. In CD-ROM, Proceedings of MEGA'2003. (2003)
8. Coddington, E., Levinson, N.: Theory of ordinary differential equations. McGraw-Hill, New York (1955)
9. Frobenius, G.: Über die Integration der linearen Differentialgleichungen mit veränder Koefficienten. Journal für die reine und angewandte Mathematik **76** (1873) 214–235
10. Heffter, L.: Einleitung in die Theorie der linearen Differentialgleichungen. Teubner, Leipzig (1894)
11. Hilali, A., Wazner, A.: Formes super–irréductibles des systèmes différentiels linéaires. Numerical Mathematics **50** (1987) 429–449
12. van der Hoeven, J.: Fast evaluation of holonomic functions near and in regular singularities. Journal of Symbolic Computation **31** (2001) 717–743
13. Kaltofen, E., Villard, G.: On the complexity of computing determinants. Computational Complexity **13** (2004) 91–130
14. Poole, E.: Introduction to the Theory of Linear Differential Equations. Dover Publications Inc., New York (1960)
15. Storjohann, A., Villard, G.: Computing the rank and a small nullspace basis of a polynomial matrix. In Proceedings of ISSAC'2005, ACM Press (2005)

Computing the Betti Numbers of Arrangements in Practice

Saugata Basu and Michael Kettner

School of Mathematics,
Georgia Institute of Technology, Atlanta, GA 30332-0160, USA
{saugata, mkettner}@math.gatech.edu

Abstract. We describe an algorithm for computing the zero-th and the first Betti numbers of the union of n simply connected compact semi-algebraic sets in $\mathbb{R}^k$, where each such set is defined by a constant number of polynomials of constant degrees. The complexity of the algorithm is $O(n^3)$. We also describe an implementation of this algorithm in the particular case of arrangements of ellipsoids in $\mathbb{R}^3$ and describe some of our results.

1 Introduction

Arrangements of geometric objects in fixed dimensional Euclidean space are fundamental objects in computational geometry [15]. Usually it is assumed that each individual object in such an arrangement has a simple description – for instance they are semi-algebraic sets defined by Boolean formulas involving a constant number of polynomial inequalities, with the degrees of the polynomials also bounded by a constant. Arrangements of semi-algebraic sets are distinguished from arrangements of lines or hyperplanes by the fact that the former can be much more complicated topologically, compared to arrangements of affine subspaces. For instance, a single algebraic hyper-surface or intersections of two or more hyper-surfaces, can have non-vanishing higher co-homology groups and thus sets defined in terms of such hyper-surfaces can be topologically complicated in various non-intuitive ways.

An important topological invariant of arrangements are the Betti numbers, $b_i(S)$, which are the ranks of $H^i(S)$ (the i-th simplicial co-homology group with coefficients in $\mathbb{Q}$). Intuitively, $b_i(S)$ measures the number of i dimensional holes in the set S. The zero-th Betti number, $b_0(S)$, is the number of connected components. For example, if S is the sphere (or an ellipsoid) in $\mathbb{R}^3$, then $b_0(S) = 1$, $b_1(S) = 0$, $b_2(S) = 1$ and $b_i(S) = 0$, $i > 2$.

There has been a significant amount of research into developing efficient algorithms for dealing with such arrangements. For instance, Chazelle, Edelsbrunner, Guibas and Sharir [11] showed how to decompose an arrangement of n objects in $\mathbb{R}^k$ into $O^*(n^{2k-3})$ simple pieces. This was further improved by Koltun in the case $k = 4$ [16]. However, these decompositions while suitable for many applications, are not useful for computing topological properties of the arrangements, since they fail to produce a cell complex. Global topological invariants of an arrangement, such as the Betti numbers, contain important information about the arrangement and computing them is an essential step in obtaining a thorough understanding of the topology of the arrangement. From a more practical point of view, the Betti numbers are increasingly being used in practice to characterize geometric objects for purposes of recognition (see for example [10]) and hence computing them in a general setting is an important problem.

The problem of computing Betti numbers of arrangements has been studied before from a computational viewpoint. Arrangements of finitely many balls in $\mathbb{R}^3$ have been studied by Edelsbrunner [13] from both combinatorial and topological viewpoint, motivated by applications in molecular biology. However, these techniques use special properties of the objects, such as convexity, and are not applicable in the more general setting considered in this paper. The problem of computing the homology groups of a given non-singular, orientable real algebraic hypersurface in $\mathbb{R}\mathrm{P}^3$, described by a single polynomial equation has been considered in [14]. However, our setting is quite different, since we are interested in computing the Betti numbers of the union of many such sets. In particular, this union will have very high degree as a real algebraic set, and will rarely be a non-singular hypersurface. Finally, the problem of computing the Betti numbers of semi-algebraic sets in single exponential time is considered to be a very important open problem in algorithmic semi-algebraic geometry. Recent progress has been made in several special cases (see [8, 5, 6]). However, in the setting of algorithmic semi-algebraic geometry one studies the dependence of the complexity on the degrees of the input polynomials, in addition to the dependence on their number. In our setting, both the degrees and the number of variables are assumed to be bounded by fixed constants. The complexity of algorithm described in this paper has a dependence on the degree d of the input polynomials in $\mathbb{R}^k$ of the form $d^{2^{O(k)}}$. However, this is still bounded by a constant by our assumptions.

One basic ingredient in most algorithms for computing topological properties of arrangements of semi-algebraic sets is an algorithm, called cylindrical algebraic decomposition [12, 7] which decomposes a given semi-algebraic set into topological balls. Cylindrical algebraic decomposition can be used to compute a semi-algebraic triangulation of an arrangement (see [7], page 163), and from this triangulation one can compute the homology groups, Betti numbers etc. One disadvantage of cylindrical algebraic decomposition is that it uses iterated projections (reducing the dimension by one in each step) and the number of polynomials (as well as the degrees) square in each step of the process. Thus, the complexity of performing cylindrical algebraic decomposition with n sets in $\mathbb{R}^k$ is $O(n^{2^k})$ which makes it impractical in most cases for computing topological information of arrangements.

One method of getting around this difficulty was suggested in [4], where an algorithm was described for computing the first ℓ Betti numbers of an arrangement using $O(n^{\ell+2})$ different triangulations each of at most $\ell+2$ of the sets at a time. This method avoids the double exponential (in the dimension) blow-up of cylindrical algebraic decomposition, and does not compute a triangulation of the whole arrangement, but yet is able to compute the Betti numbers of the arrangements. However, even computing these local triangulations can be prohibitively expensive in practice.

In this paper, we consider arrangements of compact objects in $\mathbb{R}^k$ which are simply connected. This implies, in particular, that their first Betti number is zero. We describe an algorithm for computing the zero-th and the first Betti number of such an arrangement, along with its implementation. For the implementation, we restrict our attention to arrangements in $\mathbb{R}^3$ and take for our objects the simplest possible semi-algebraic sets in $\mathbb{R}^3$ which are topologically non-trivial – namely, each object is an ellipsoid defined by a single quadratic equation. Ellipsoids are simply connected, but with non-vanishing second co-homology groups. We also allow solid ellipsoids defined by a single quadratic inequality. Computing the Betti numbers of an arrangement of ellipsoids in $\mathbb{R}^3$ is already a challenging computational problem in practice and to our knowledge no existing software can effectively deal with this case. Note that arrangements of ellipsoids are topologically quite different from arrangements of balls. For instance, the union of two ellipsoids can have non-zero first Betti number, unlike in the case of balls.

In our algorithm we do not need to compute any triangulations. Instead, we use cylindrical algebraic decompositions to identify the connected components of the pairwise and triple-wise intersections of the input ellipsoids. For computing these local cylindrical algebraic decompositions we use a package, QEPCAD B[1], which is currently the one of the best (for our needs) available packages for computing cylindrical algebraic decomposition. We note that in our examples QEPCAD B fails to produce a global cylindrical algebraic decomposition, while it can produce all the local ones that we need.

Theoretically, our algorithm is similar to the one described in [8] for computing the first Betti numbers of semi-algebraic sets in single exponential time. The main difference is that we do not compute contractible covers. Instead, we utilize the fact that the first co-homology groups of the input objects are trivial. In order to prove the correctness of our algorithm, we show that the main mathematical result behind correctness of the algorithm in [8] is in fact valid with the weaker assumption that the elements of the cover have vanishing co-homology in dimension one (see Proposition 1 below). We include several examples of our computations involving up to 20 ellipsoids in $\mathbb{R}^3$.

The main contribution of this paper is the following: We describe an algorithm for computing the zero-th and the first Betti numbers of the union of n simply connected compact semi-algebraic sets in $\mathbb{R}^k$. The algorithm does not use triangulations. The complexity of the algorithm is $O(n^3)$. We also describe an implementation of this algorithm in the particular case of arrangements of ellipsoids in $\mathbb{R}^3$.

The rest of the paper is organized as follows. In Section 2, we describe the theoretical and practical techniques used to obtain our results. In Section 3, we

[1] available at `http://www.cs.usna/∼qepcad`

describe some results obtained using our implementation, and finally in Section 4, we summarize our current and future work.

2 Techniques

2.1 Topological Preliminaries

We first prove a topological result that will play a fundamental role in our algorithm.

Let $A_1, \ldots, A_n$ be sub-complexes of a finite simplicial complex A such that $A = A_1 \cup \cdots \cup A_n$. Note that the intersections of any number of the sub-complexes, A_i, is again a sub-complex of A. We will denote by $A_{i_0,\ldots,i_p}$ the sub-complex $A_{i_0} \cap \cdots \cap A_{i_p}$.

Let $C^i(A)$ denote the $\mathbb{Q}$-vector space of i co-chains of A, and $C^\bullet(A)$, the complex

$$\cdots \rightarrow C^{q-1}(A) \xrightarrow{d} C^q(A) \xrightarrow{d} C^{q+1}(A) \rightarrow \cdots$$

where $d : C^q(A) \rightarrow C^{q+1}(A)$ are the usual co-boundary homomorphisms. More precisely, given $\omega \in C^q(A)$, and a $q+1$ simplex $[a_0, \ldots, a_{q+1}] \in A$,

$$d\omega([a_0, \ldots, a_{q+1}]) = \sum_{0 \le i \le q+1} (-1)^i \omega([a_0, \ldots, \hat{a_i}, \ldots, a_{q+1}]) \qquad (1)$$

(here and everywhere else in the paper $\hat{}$ denotes omission). Now extend $d\omega$ to a linear form on all of $C^{q+1}(A)$ by linearity, to obtain an element of $C^{q+1}(A)$.

The generalized Mayer-Vietoris sequence is the following:

$$0 \longrightarrow C^\bullet(A) \xrightarrow{r} \prod_{i_0} C^\bullet(A_{i_0}) \xrightarrow{\delta_1} \prod_{i_0<i_1} C^\bullet(A_{i_0,i_1})$$

$$\cdots \xrightarrow{\delta_{p-1}} \prod_{i_0<\cdots<i_p} C^\bullet(A_{i_0,\ldots,i_p}) \xrightarrow{\delta_p} \prod_{i_0<\cdots<i_{p+1}} C^\bullet(A_{i_0,\ldots,i_{p+1}}) \cdots$$

where r is induced by restriction and the connecting homomorphisms δ are described below.

Given an $\omega \in \prod_{i_0<\cdots<i_p} C^q(A_{i_0,\ldots,i_p})$ we define $\delta(\omega)$ as follows:
First note that $\delta(\omega) \in \prod_{i_0<\cdots<i_{p+1}} C^q(A_{i_0,\ldots,i_{p+1}})$, and it suffices to define $\delta(\omega)_{i_0,\ldots,i_{p+1}}$ for each $(p+2)$-tuple $0 \le i_0 < \cdots < i_{p+1} \le n$. Note that, $\delta(\omega)_{i_0,\ldots,i_{p+1}}$ is a linear form on the vector space, $C_q(A_{i_0,\ldots,i_{p+1}})$, and hence is determined by its values on the q-simplices in the complex $A_{i_0,\ldots,i_{p+1}}$. Furthermore, each q-simplex, $s \in A_{i_0,\ldots,i_{p+1}}$ is automatically a simplex of the complexes $A_{i_0,\ldots,\hat{i_i},\ldots i_{p+1}}$, $0 \le i \le p+1$.

We define,

$$(\delta\omega)_{i_0,\ldots,i_{p+1}}(s) = \sum_{0 \le i \le p+1} (-1)^i \omega_{i_0,\ldots,\hat{i_i},\ldots,i_{p+1}}(s),$$

The fact that the generalized Mayer-Vietoris sequence is exact is classical (see [4] for example).

The co-homology groups $H^0(A_{i_0,\dots,i_p})$ are isomorphic to the $\mathbb{Q}$-vector space of locally constant functions on $A_{i_0,\dots,i_p}$ and the induced homomorphisms, $\delta_p : H^*(A_{i_0,\dots,i_p}) \to H^*(A_{i_0,\dots,i_{p+1}})$ are then given by generalized restrictions, i.e. for

$$\phi \in \oplus_{1\le i_0<\cdots<i_p\le n} H^0(A_{i_0,\dots,i_p}),$$

where each $\phi_{i_0,\dots,i_p}$ is a locally constant function on $A_{i_0,\dots,i_p}$,

$$\delta_p(\phi)_{i_0,\dots,i_{p+1}} = \sum_{i=0}^{p} (-1)^i \phi_{i_0,\dots,\hat{i_i},\dots,i_{p+1}}|_{A_{i_0,\dots,i_{p+1}}}.$$

The following proposition is a slightly strengthened version of a similar proposition appearing in [8]. We do not require that the complexes A_i be acyclic, but only that their first co-homology group vanishes.

Proposition 1. *Let $A_1,\dots,A_n$ be sub-complexes of a finite simplicial complex A such that $A = A_1 \cup \cdots \cup A_n$ and for each $i, 1 \le i \le n$,*

1. $H^0(A_i) = \mathbb{Q}$, and
2. $H^1(A_i) = 0$.

Let the homomorphisms δ_1, δ_2 in the following sequence be defined as above.

$$\prod_i H^0(A_i) \xrightarrow{\delta_1} \prod_{i<j} H^0(A_{i,j}) \xrightarrow{\delta_2} \prod_{i<j<\ell} H^0(A_{i,j,\ell})$$

Then,

1. $b_0(A) = \dim(\mathrm{Ker}(\delta_1))$,
2. $b_1(A) = \dim(\mathrm{Ker}(\delta_2)) - \dim(\mathrm{Im}(\delta_1))$.

Proof. See Appendix.

As a direct corollary of the above proposition we have,

Corollary 1. *Let $S_1,\dots,S_n \subset \mathbb{R}^k$ be compact semi-algebraic sets such that for each $i, 1 \le i \le n$,*

1. $H^0(S_i) = \mathbb{Q}$, and
2. $H^1(S_i) = 0$.

Let the homomorphisms δ_1, δ_2 in the following sequence be defined as above (identifying $H^0(X)$ with the $\mathbb{Q}$-vector space of locally constant functions on X).

$$\prod_i H^0(S_i) \xrightarrow{\delta_1} \prod_{i<j} H^0(S_{i,j}) \xrightarrow{\delta_2} \prod_{i<j<\ell} H^0(S_{i,j,\ell})$$

Then,

1. $b_0(S) = \dim(\mathrm{Ker}(\delta_1))$,
2. $b_1(S) = \dim(\mathrm{Ker}(\delta_2)) - \dim(\mathrm{Im}(\delta_1))$.

Remark 1. Recall that we only consider simply connected objects as our input sets S_i. Hence, we have $b_0(S_i) = 1$ and $b_1(S_i) = 0$ for all sets S_i. In particular, we consider arrangements of ellipsoids $\mathbb{R}^3$ which are simply connected as well. Moreover, the assumption that $b_0(S_i) = 1$ is not necessary, but simplifies the implementation.

The importance of Corollary 1 lies in the following observation. Given an arrangement, $\{S_1, \dots, S_n\}$, of n simply connected objects in $\mathbb{R}^k$, suppose we are able to identify the connected components of all pairwise and triple-wise intersections of these objects and their incidences (that is, which connected component of $S_i \cap S_j \cap S_\ell$ is contained in which connected component of $S_i \cap S_j$). Then this information is sufficient to compute the zero-th and the first Betti number of the arrangement. We only have to look at the objects of the arrangement at most three at a time. Thus, the cost of computing the connected components and incidences is $O(n^3)$. This is to be compared with having to compute a global triangulation of the whole arrangement using cylindrical algebraic decomposition which would have entailed a cost of $O(n^{2^k})$.

We solve the problem of identifying the connected components of the pairwise and triple-wise intersections using cylindrical algebraic decomposition, which we describe next.

2.2 Cylindrical Algebraic Decomposition

In this section, we recall some facts about cylindrical algebraic decomposition, which is used to identify the connected components of the pairwise and triple-wise intersections of the input ellipsoids. For more details on this subject, see [7, 1, 2, 3].

Definition 1. *A* ***Cylindrical Algebraic Decomposition*** *(CAD) of $\mathbb{R}^k$ is a sequence $\mathcal{S}_1, \dots, \mathcal{S}_k$, where, for each $1 \le i \le k$, $\mathcal{S}_i$ is a finite partition of $\mathbb{R}^i$ into semi-algebraic subsets (**cells of level i**), which satisfy the following properties:*

Each cell $C \in \mathcal{S}_1$ is either a point or an open interval.
For every $1 \le i < k$ and every $C \in \mathcal{S}_i$ there are finitely many continuous semi-algebraic functions

$$\xi_{C,1} < \cdots < \xi_{C,\ell_C} : C \to \mathbb{R}$$

*such that the **cylinder** $C \times \mathbb{R} \subset \mathbb{R}^{i+1}$ (also called a **stack over the cell C**) is a disjoint union of cells of $\mathcal{S}_{i+1}$ which are:*
either the graph of one of the functions $\xi_{C,j}$, for $j = 1, \dots, \ell_C$:

$$\{(x', x_{j+1}) \in C \times \mathbb{R} | x_{j+1} = \xi_{C,j}(x')\},$$

or a band of the cylinder bounded from below and above by the graphs of the functions $\xi_{C,j}$ and $\xi_{C,j+1}$, for $j = 0, \dots, \ell_C$, where we take $\xi_{C,0} = -\infty$ and $\xi_{C,\ell_C+1} = +\infty$.

Note that a cylindrical algebraic decomposition has a recursive structure, i.e. the decomposition of $\mathbb{R}^i$ induces a decomposition of $\mathbb{R}^{i+1}$ and vice-versa.

Definition 2. *Given a finite set $\mathcal{P}$ of polynomials in $\mathbb{R}[X_1, \ldots, X_k]$, a subset S of $\mathbb{R}^k$ is **$\mathcal{P}$-invariant** if every polynomial P in $\mathcal{P}$ has constant sign on S. A **cylindrical decomposition of** $\mathbb{R}^k$ **adapted to** $\mathcal{P}$ is a cylindrical algebraic decomposition for which each cell $C \in \mathcal{S}_k$ is $\mathcal{P}$-invariant.*

The following example illustrate the above definitions.

Example 1 (CAD adapted to the unit sphere). Let be $S = \{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2+z^2-1 = 0\}$ (see figure 1). The decomposition of $\mathbb{R}$ (i.e. the line) consists of five cells of level 1 corresponding to the points -1 and 1 and the three intervals they define. The decomposition of $\mathbb{R}^2$ (i.e., the plane) consists of 13 cells of level 2. For instance, the two bands to the left and right of the circle, the two cells corresponding to the points $(-1, 0)$ and $(1, 0)$ and the cell that corresponds to the set $C_{3,2} = \{(x, y) \mid -1 < x < 1, y = -\sqrt{1 - x^2}\}$. The CAD of $\mathbb{R}^3$ consists of 25 cells of level 3. For instance, the two cells corresponding to the points $(-1, 0, 0)$ and $(1, 0, 0)$ and the cell that corresponds to the set $C_{3,2,2} = C_{3,2} \times \{0\}$. For a more detailed description of this example see [7], Chapter 5.1.

An important piece of information that we require from the cylindrical algebraic decomposition algorithm is that of **cell adjacency** [3]. In other words, we need to know given two cells in a set $\mathcal{S}_i$, whether the closure of one intersects the other. In the above example, for instance, we have that the cell corresponding to the point $(-1, 0, 0)$ is adjacent to the cell $C_{3,2,2}$. More details on this matter are given in Section 2.3.

Thus, via a cylindrical algebraic decomposition adapted to a finite set $\mathcal{P}$ of polynomials in $\mathbb{R}[X_1, \ldots, X_k]$ we get a graph where the vertices correspond to cells in $\mathcal{S}_k$ and edges correspond to adjacencies. Moreover, each cell in $\mathcal{S}_k$ is $\mathcal{P}$-invariant and we know the sign for each P in $\mathcal{P}$ on each such cell. Hence, given an

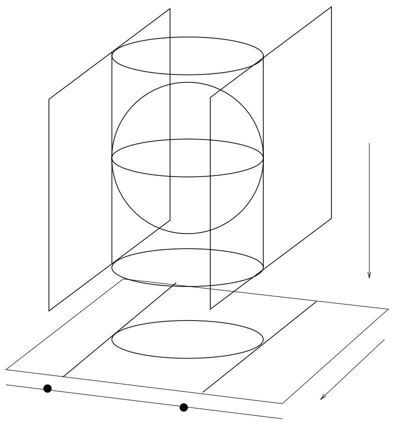


Fig. 1. A cylindrical algebraic decomposition adapted to the unit sphere in $\mathbb{R}^3$

arrangement, $\{S_1, \dots, S_n\}$, of n semi-algebraic sets in $\mathbb{R}^k$, we are able to identify the connected components of all pairwise and triple-wise intersections of these objects and their incidences by computing a cylindrical algebraic decomposition adapted to the families $\mathcal{P}_{i,j,\ell}$, $1 \le i < j < \ell \le n$, where $\mathcal{P}_{i,j,\ell}$ is the set of polynomials used in the definition of S_i, S_j, and S_ℓ and by performing a graph transversal algorithm on the graph described above.

We now formally describe our algorithm for computing the zero-th and the first Betti numbers of an arrangement of n simply connected compact objects in $\mathbb{R}^k$.

Algorithm 1 (Computing the zero-th and the first Betti number)
Input: *compact sets* $S_i \subset \mathbb{R}^k$, $1 \le i \le n$, *with* $b_0(S_i) = 1$ *and* $b_1(S_i) = 0$.
Output: $b_0(S)$ *and* $b_1(S)$.
Procedure:

```
Step 1: For each triple $(i, j, \ell)$, $1 \le i < j < \ell \le n$, do the following:
            Compute a CAD adapted to the set $\{S_i, S_j, S_\ell\}$.
            Identify the connected components of all pairwise and triple-wise intersections and their incidences.
Step 2: Compute the matrices $A$ and $B$ corresponding to the sequence of homomorphisms:
```

$$\bigoplus_i H^0(S_i) \xrightarrow{\delta_1} \bigoplus_{i<j} H^0(S_i \cap S_j) \xrightarrow{\delta_2} \bigoplus_{i<j<\ell} H^0(S_i \cap S_j \cap S_\ell).$$

```
Step 3: Compute
```

$$b_0(S) = d_0 - \mathrm{rk}(A), \qquad (2)$$

and

$$b_1(S) = d_1 - \mathrm{rk}(B) - \mathrm{rk}(A), \qquad (3)$$

where d_0 *is the dimension of* $\bigoplus_{1 \le i \le n} H^0(S_i)$, d_1 *is the dimension of* $\bigoplus_{1 \le i < j \le n} H^0(S_i \cap S_j)$, *and the rank of a matrix is denoted by* $\mathrm{rk}(\cdot)$.

2.3 Practical Techniques

As noted previously, we use the package QEPCAD B (Version 1.27) for computing the cylindrical algebraic decompositions, in Step 1 of Algorithm 1. There are several other packages available for computing cylindrical algebraic decompositions, for instance REDLOG[2]. The main reason for using QEPCAD B is that it provides some important information regarding cell adjacency, that is not provided by the other systems. Even though QEPCAD B does not yet provide full information regarding cell adjacencies in dimension three, we are still able to deduce all the needed cell adjacencies, making use of the fact that input polynomials are quadratic. We use another system, Magma[3] for post-processing of the information output by QEPCAD B, in Steps 2 and 3 of the algorithm.

[2] available at `http://www.fmi.uni-passau.de/~redlog`
[3] available at `http://magma.maths.usyd.edu.au/magma`

Note that all computations performed are exact with no possibility of numerical errors.

In order to describe our method for obtaining the needed cell adjacencies for a cylindrical algebraic composition adapted to a finite set $\mathcal{P}$ of polynomials, we need the following notation.

We distinguish between the **inter-stack cell adjacency of level** i, which is the adjacency of cells of level i in two different stacks, and the **intra-stack cell adjacency of level** i, which is the adjacency of cells of level i within the same stack.

Moreover, we use the same intuitive labeling of cells as used by QEPCAD B which we describe next.

A cell in $\mathbb{R}$, i.e. a cell in the induced CAD (line) of the induced CAD (plane), is denoted by (i), where the i ranges over the number of cells in the induced CAD of $\mathbb{R}$. Note that $i_1 < i_2$ if and only if the cell (i_1) "occurs to the left" of the cell (i_2).

A cell in $\mathbb{R}^2$, i.e. a cell in the induced CAD of the plane, is denoted by (i, j), where i ranges over the number of cells in the line and the j ranges over the number of cells in the stack over the cell (i). Note that $j_1 < j_2$ if and only if the cell (i, j_1) "occurs lower in the plane" than the cell (i, j_2).

A cell in $\mathbb{R}^3$ is denoted by (i, j, k), where (i, j) is a cell in the induced CAD of the plane and the k ranges over the number of cells in the stack over the cell (i, j). Note that $k_1 < k_2$ if and only if the cell (i, j, k_1) "occurs lower" than the cell (i, j, k_2).

Furthermore, we distinguish among **0-cells, 1-cells, 2-cells** and **3-cells** of the CAD, that are points, graphs and cylinders bounded below and above by graphs. The adjacency between a ℓ-cell and k-cell will be denoted by $\{\ell, \mathbf{k}\}$**-adjacency**.

We illustrate the above notation on Example 1 (CAD adapted to the unit sphere)

Example 2 (cont.). For instance, the cell (2) and (4) correspond to the points -1 and 1 (in the line), whereas the cells $(2, 2)$ and $(3, 2)$ correspond the point $(-1, 0)$ and the set $C_{3,2} = \{(x, y) \mid -1 < x < 1, y = -\sqrt{1 - x^2}\}$. Moreover, the cell $(2, 2, 2)$ corresponds to the point $(-1, 0, 0)$ and the cell $(3, 2, 2)$ corresponds to the set $C_{3,2,2} = C_{3,2} \times \{0\}$.

In Figure 2, which shows the equivalent QEPCAD B output, the first (resp., second and third) column corresponds to the CAD of the line (resp. plane and $\mathbb{R}^3$). Note that the signs accompanying the cells give the signs of projection factors computed by QEPCAD B and the letter "T" and "F" corresponds to true and false value of the cells, i.e. depending upon whether our input formula is true or false on this cell.

The following cell adjacency information is either provided by QEPCAD B or can easily be determined:

1. The intra-stack adjacency information. By labeling the cells in each stack from the "bottom" to the "top" we get the the intra-stack adjacency information on every level. For instance, the cell (i, j) is adjacent to the cell

```
()---(1)p1(-,-)---(1,1)p1(+)---(1,1,1)p1(+) F
  ---(2)p1(0,-)---(2,1)p1(+)---(2,1,1)p1(+) F
               ---(2,2)p1(0)---(2,2,1)p1(+) F
                            ---(2,2,2)p1(0) T
                            ---(2,2,3)p1(+) F
               ---(2,3)p1(+)---(2,3,1)p1(+) F
  ---(3)p1(+,-)---(3,1)p1(+)---(3,1,1)p1(+) F
               ---(3,2)p1(0)---(3,2,1)p1(+) F
                            ---(3,2,2)p1(0) T
                            ---(3,2,3)p1(+) F
               ---(3,3)p1(-)---(3,3,1)p1(+) F
                            ---(3,3,2)p2(0) T
                            ---(3,3,3)p1(-) F
                            ---(3,3,4)p2(0) T
                            ---(3,3,5)p1(+) F
               ---(3,4)p1(0)---(3,4,1)p1(+) F
                            ---(3,4,2)p1(0) T
                            ---(3,4,3)p1(+) F
               ---(3,5)p1(+)---(3,5,1)p1(+) F
  ---(4)p1(+,0)---(4,1)p1(+)---(4,1,1)p1(+) F
               ---(4,2)p1(0)---(4,2,1)p1(+) F
                            ---(4,2,2)p1(0) T
                            ---(4,2,3)p1(+) F
               ---(4,3)p1(+)---(4,3,1)p1(+) F
  ---(5)p1(+,+)---(5,1)p1(+)---(5,1,1)p1(+) F
```

Fig. 2. Output of a CAD adapted to the unit sphere using QEPCAD B

$(i, j+1)$ (in the induced CAD of the plane) or the cell (i, j, k) is adjacent to the cell $(i, j, k+1)$

2. The {0,1}-inter-stack adjacency information of level 2. Together with the intra-stack adjacency information we could determine all the other missing adjacencies for the induced CAD of the plane, although it is not necessary for our application.
3. The inter-stack adjacency information of level 3, if the induced cells are within the same (induced) stack and the cells are part of the boundary of the semi-algebraic sets S_i. In other words, we get the adjacency information for the stacks of the cells (i, j) and $(i, j+1)$.

Example 3 (cont.). For our example above, we get the following cell adjacency information for the induced CAD: e.g.
intra-stack: ((3,2),(3,3)), inter-stack: ((2,2),(3,2))
and for the CAD of $\mathbb{R}^3$:
intra-stack: ((3,3,2),(3,3,3)), inter-stack: ((3,2,2),(3,3,2)) missing: ((2,2,2), (3,2,2))

Unfortunately, there is also some missing adjacency information. For instance, assuming that in the induced CAD of the plane the cells (i, j_1) and $(i+1, j_2)$ are adjacent, then we do not get (directly) any adjacency information for the corresponding stacks. But as we see next, we are able to construct the missing adjacency information. It is worthwhile to mention that we do not determine the whole inter-stack adjacency information for the complete CAD. Instead, we determine the full adjacency information for the boundary of the semi-algebraic set defined by the input formula.

Recall that we consider only balls, ellipsoids and spheres as our semi algebraic input sets S_i , i.e., $S_i = \{(x, y, z) \in \mathbb{R}^3 \mid P_i(x, y, z) * 0\}$ where $deg(P_i(X, Y, Z)) = 2$ and $* \in \{\leq, =\}$.

Moreover, assume that the 0-cell (i, j_1) and the 1-cell $(i + 1, j_2)$ are adjacent in the induced CAD of the plane. We have the following two cases:

Case 1: The stack over the 0-cell (i, j_1) contains exactly one 0-cell (i, j_1, k). Note, that the stack over 1-cell $(i + 1, j_2)$ must contain two 1-cells $(i + 1, j_2, l_1)$ and $(i + 1, j_2, l_2)$ (corresponding to graphs), since the polynomial P_i is of total degree equal to 2. Therefore, the 0-cell (i, j_1, k) must be adjacent to both cells $(i + 1, j_2, l_1)$ and $(i + 1, j_2, l_2)$, since the semi algebraic set S_i is closed.

Case 2: The stack over the 0-cell (i, j_1) contains two 0-cells (i, j_1, k_1) and (i, j_1, k_2). As above, the stack over the 1-cell $(i + 1, j_2)$ must contain two 1-cells $(i + 1, j_2, \ell_1)$ and $(i + 1, j_2, \ell_2)$. Remember that, both stacks are ordered from the bottom to the top. Hence, the cells (i, j_1, k_1) and $(i + 1, j_2, \ell_1)$ as well as the cells (i, j_1, k_2) and $(i + 1, j_2, \ell_2)$ must be adjacent for the same reason as above.

Remark 2. Although there are several algorithms known for computing complete cell adjacency information (see [2] and [3]), no algorithm is completely implemented at the moment. Moreover, constructing the missing adjacency information as described above would fail for polynomials of total degree greater than two.

3 Examples

To illustrate our implementation, we consider four examples where the ellipsoids $S_i = \{(x, y, z) \in \mathbb{R}^3 \mid P_i(x, y, z) = 0\}$, $1 \leq i \leq 27$, are defined by the following list of polynomials.

$$\begin{aligned}
P_1 &= 8/9X^2 + 1/64Y^2 + 1/6Z^2 - 1\\
P_2 &= 1/64X^2 + 8/9Y^2 + 8/9Z^2 - 1\\
P_3 &= 8/9X^2 + 8/9Y^2 + 1/64Z^2 - 1\\
P_4 &= 8/9(X-4)^2 + 1/64(Y-4)^2 + 1/6Z^2 - 1\\
P_5 &= 1/64(X-4)^2 + 8/9(Y-4)^2 + 8/9Z^2 - 1\\
P_6 &= 8/9(X-4)^2 + 8/9(Y-4)^2 + 1/64Z^2 - 1\\
P_7 &= (X-1)^2 + (Y-2)^2 + Z^2 - 3
\end{aligned}$$

$$\begin{aligned}
P_8 &= 5X^2 + 1/9Y^2 + 2Z^2 - 1\\
P_9 &= 1/9X^2 + 5Y^2 + 5Z^2 - 1\\
P_{10} &= 5X^2 + 5Y^2 + 1/9Z^2 - 1\\
P_{11} &= 5(X-1)^2 + 1/9(Y-1)^2 + 2Z^2 - 1\\
P_{12} &= 1/9(X-1)^2 + 5(Y-1)^2 + 5Z^2 - 1\\
P_{13} &= 5(X-1)^2 + 5(Y-1)^2 + 1/9Z^2 - 1\\
P_{14} &= 5(X+1)^2 + 1/9(Y-1)^2 + 2Z^2 - 1\\
P_{15} &= 1/9(X+1)^2 + 5(Y-1)^2 + 5Z^2 - 1\\
P_{16} &= 5(X+1)^2 + 5(Y-1)^2 + 1/9Z^2 - 1\\
P_{17} &= 5(X-1)^2 + 1/9(Y+1)^2 + 2Z^2 - 1\\
P_{18} &= 1/9(X-1)^2 + 5(Y+1)^2 + 5Z^2 - 1\\
P_{19} &= 5(X-1)^2 + 5(Y+1)^2 + 1/9Z^2 - 1\\
P_{20} &= 5(X+1)^2 + 1/9(Y+1)^2 + 2Z^2 - 1\\
P_{21} &= 1/9(X+1)^2 + 5(Y+1)^2 + 5Z^2 - 1\\
P_{22} &= 5(X+1)^2 + 5(Y+1)^2 + 1/9Z^2 - 1\\
P_{23} &= 6(X-1/2)^2 + 6Y^2 + 1/6Z^2 - 1\\
P_{24} &= 4X^2 + 4(Y-1/2)^2 + 1/6Z^2 - 1\\
P_{25} &= 5(X+2)^2 + 5Y^2 + 1/6Z^2 - 1\\
P_{26} &= 1/6(X+2)^2 + 5(Y-2)^2 + 5Z^2 - 1\\
P_{27} &= 5(X+2)^2 + 1/6(Y-2)^2 + 5Z^2 - 1
\end{aligned}$$

We denote by A and B the matrices of the homomorphisms δ_1 and δ_2 with respect to the obvious basis. The columns (respectively, the rows) of the matrix A are labeled by e_i (respectively, $e^p_{i,j}$), while the columns (respectively, the rows) of the matrix B are labeled by $e^p_{i,j}$ (respectively, $e^p_{i,j,k}$), where e_i corresponds to S_i, $e^p_{i,j}$ corresponds to the p-th connected component of $S_i \cap S_j$ and $e^p_{i,j,\ell}$ corresponds to the p-th connected component of $S_i \cap S_j \cap S_\ell$.

Remark 3. In the examples described below, we have modified the matrix A as follows. Since we know that each input set S_i has exactly one connected component, we can simplify the computation. We only need to check whether or not the intersection $S_i \cap S_j$ is empty. Therefore, we have exactly one row for each intersection instead of one row for each connected component of each intersection $S_i \cap S_j$, and this reduces the size of the matrix A without changing its rank. For the matrix B we delete all rows containing only zeros which correspond to empty triple intersections $S_i \cap S_j \cap S_\ell$.

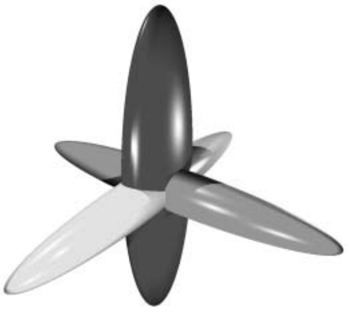


Fig. 3. three ellipsoids

Example 4 (three ellipsoids). Let S be the union of the first three ellipsoids, i.e. $S = S_1 \cup S_2 \cup S_3$ (see Figure 3). Then,

$$A = \left(\begin{array}{ccc} e_1 & e_2 & e_3 \\ \hline -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{array}\right) \begin{array}{c} \\ e_{1,2} \\ e_{1,3} \\ e_{2,3} \end{array} \text{ and } B = \left(\begin{array}{cccc} e^1_{1,2} & e^2_{1,2} & e_{1,3} & e_{2,3} \\ \hline 1 & 0 & -1 & 1 \\ 1 & 0 & -1 & 1 \\ 1 & 0 & -1 & 1 \\ 1 & 0 & -1 & 1 \\ 0 & 1 & -1 & 1 \\ 0 & 1 & -1 & 1 \\ 0 & 1 & -1 & 1 \\ 0 & 1 & -1 & 1 \end{array}\right) \begin{array}{c} \\ e^1_{1,2,3} \\ e^2_{1,2,3} \\ e^3_{1,2,3} \\ e^4_{1,2,3} \\ e^5_{1,2,3} \\ e^6_{1,2,3} \\ e^7_{1,2,3} \\ e^8_{1,2,3} \end{array}$$

In this case,

$$b_0(S) = d_0 - \mathrm{rk}(A) = 3 - 2 = 1$$
$$b_1(S) = d_1 - \mathrm{rk}(B) - \mathrm{rk}(A) = (4 - 2) - 2 = 0$$

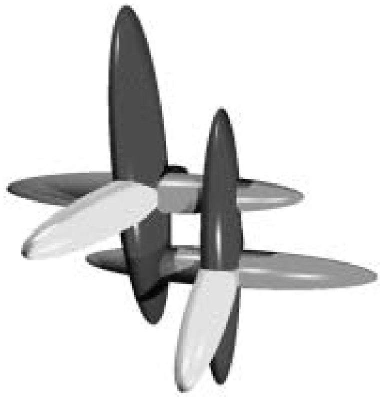


Fig. 4. six ellipsoids

Example 5 (six ellipsoids). Let the set S be the union of the first six ellipsoids S_i, $1 \leq i \leq 6$, i.e. $S = S_1 \cup \ldots \cup S_6$ (see Figure 4). Then,

$$A = \begin{array}{c} \left(\begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \hline -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right) \end{array} \begin{array}{c} \\ e_{1,2} \\ e_{1,3} \\ e_{1,4} \\ e_{1,5} \\ e_{1,6} \\ e_{2,3} \\ e_{2,4} \\ e_{2,5} \\ e_{2,6} \\ e_{3,4} \\ e_{3,5} \\ e_{3,6} \\ e_{4,5} \\ e_{4,6} \\ e_{5,6} \end{array}$$

and

$$B = \left(\begin{array}{cccccccccccc} e^1_{1,2} & e^2_{1,2} & e^1_{1,3} & e^1_{1,5} & e^2_{1,5} & e^1_{2,3} & e^1_{2,4} & e^2_{2,4} & e^1_{4,5} & e^2_{4,5} & e^1_{4,6} & e^1_{5,6} \\ \hline 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{array}\right) \begin{array}{c} \\ e^1_{1,2,3} \\ e^2_{1,2,3} \\ e^3_{1,2,3} \\ e^4_{1,2,3} \\ e^5_{1,2,3} \\ e^6_{1,2,3} \\ e^7_{1,2,3} \\ e^8_{1,2,3} \\ e^1_{4,5,6} \\ e^2_{4,5,6} \\ e^3_{4,5,6} \\ e^4_{4,5,6} \\ e^5_{4,5,6} \\ e^6_{4,5,6} \\ e^7_{4,5,6} \\ e^8_{4,5,6} \end{array}$$

In this case,

$$b_0(S) = d_0 - \mathrm{rk}(A) = 6 - 5 = 1$$

$$b_1(S) = d_1 - \mathrm{rk}(B) - \mathrm{rk}(A) = (12 - 4) - 5 = 3$$

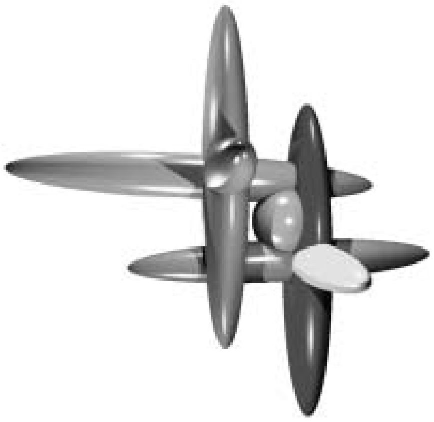


Fig. 5. seven ellipsoids

Example 6 (seven ellipsoids). Let the set S be the union of the first seven ellipsoids S_i, $1 \le i \le 7$, i.e. $S = S_1 \cup \ldots \cup S_7$ (see Figure 5). Then,

$$A = \left(\begin{array}{ccccccc|c}
e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & \\
\hline
-1 & 1 & 0 & 0 & 0 & 0 & 0 & e_{1,2} \\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & e_{1,3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{1,4} \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & e_{1,5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{1,6} \\
-1 & 0 & 0 & 0 & 0 & 0 & 1 & e_{1,7} \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & e_{2,3} \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & e_{2,4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{2,5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{2,6} \\
0 & -1 & 0 & 0 & 0 & 0 & 1 & e_{2,7} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{3,4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{3,5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{3,6} \\
0 & 0 & -1 & 0 & 0 & 0 & 1 & e_{3,7} \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & e_{4,5} \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & e_{4,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{4,7} \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & e_{5,6} \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & e_{5,7} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{6,7}
\end{array}\right)$$

and

$$B = \left(\begin{array}{cccccccccccccccc|c}
e^1_{1,2} & e^2_{1,2} & e^1_{1,3} & e^1_{1,5} & e^2_{1,5} & e^1_{1,7} & e^1_{2,3} & e^1_{2,4} & e^2_{2,4} & e^1_{2,7} & e^1_{3,7} & e^1_{4,5} & e^2_{4,5} & e^1_{4,6} & e^1_{5,6} & e^1_{5,7} & \\
\hline
1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^1_{1,2,3} \\
1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^2_{1,2,3} \\
1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^3_{1,2,3} \\
1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^4_{1,2,3} \\
0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^5_{1,2,3} \\
0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^6_{1,2,3} \\
0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^7_{1,2,3} \\
0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^8_{1,2,3} \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & e^1_{1,2,7} \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & e^2_{1,2,7} \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & e^1_{1,5,7} \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & e^2_{1,5,7} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & e^1_{2,3,7} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & e^2_{2,3,7} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & e^3_{2,3,7} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & e^4_{2,3,7} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & e^1_{4,5,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & e^2_{4,5,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & e^3_{4,5,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & e^4_{4,5,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & e^5_{4,5,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & e^6_{4,5,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & e^7_{4,5,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & e^8_{4,5,6}
\end{array}\right)$$

In this case,

$$b_0(S) = d_0 - \mathrm{rk}(A) = 7 - 6 = 1$$

$$b_1(S) = d_1 - \mathrm{rk}(B) - \mathrm{rk}(A) = (16 - 7) - 6 = 3$$

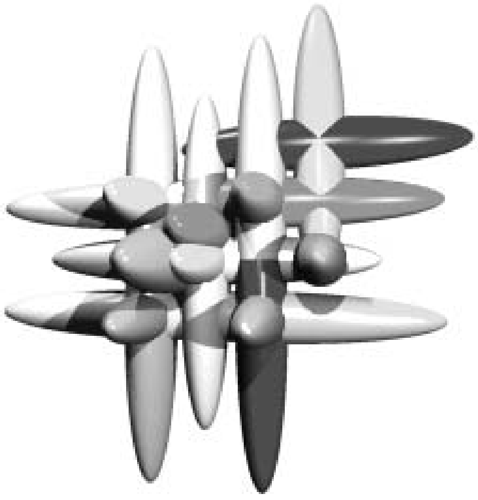


Fig. 6. twenty ellipsoids

Example 7 (20 ellipsoids). Let the set S be the union of the last 20 ellipsoids S_i, $8 \leq i \leq 27$, i.e. $S = S_8 \cup \ldots \cup S_{27}$ (see Figure 6). Thus, we get a 190×20-matrix A of rank equal to 19, a 190×107-matrix B of rank equal to 55, and the dimension of $\bigoplus_{i<j} H^0(S_i \cap S_j)$ is equal to 107. In this case,

$$b_0(S) = d_0 - \mathrm{rk}(A) = 20 - 19 = 1$$
$$b_1(S) = d_1 - \mathrm{rk}(B) - \mathrm{rk}(A) = 107 - 55 - 19 = 33$$

4 Conclusion and Future Work

We have shown that it is feasible to compute topological invariants of arrangements of semi-algebraic sets using exact computations, provided one uses algorithms with reasonable combinatorial complexity. In the future, we hope to enlarge the class of objects that we can deal with. This would require improvement in the CAD software packages – namely implementation of the cell adjacency algorithms. We hope to be able to handle polynomials of degree up to four, once complete cell adjacency information is available in QEPCAD B.

Acknowledgment

The first author was supported in part by an NSF Career Award 0133597 and an Alfred P. Sloan Foundation Fellowship. Parts of the work were completed while the second author held a position at the University of Cantabria in Santander (Spain) supported by the European RTNetwork Real Algebraic and Analytic Geometry, Contract No. HPRN-CT-2001-00271.

The authors thank Christoper W. Brown for his steady help with QEPCAD B.

References

[1] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition. I. The basic algorithm. *SIAM J. Comput.*, 13(4):865–877, 1984.

[2] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition. II. An adjacency algorithm for the plane. *SIAM J. Comput.*, 13(4):878–889, 1984.

[3] D. S. Arnon, G. E. Collins, and S. McCallum. An adjacency algorithm for cylindrical algebraic decompositions of three-dimensional space. *J. Symbolic Comput.*, 5(1-2):163–187, 1988.

[4] S. Basu. Computing the Betti numbers of arrangements via spectral sequences. *J. Comput. System Sci.*, 67(2):244–262, 2003. Special issue on STOC2002 (Montreal, QC).

[5] S. Basu. Computing the first few Betti numbers of semi-algebraic sets in single exponential time. Preprint, 2005.

[6] S. Basu. Computing the top Betti numbers of semi-algebraic sets defined by quadratic inequalities in polynomial time. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, 2005.

[7] 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.

[8] S. Basu, R. Pollack, and M.-F. Roy. Computing the first Betti number and the connected components of semi-algebraic sets. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, 2005.

[9] C. W. Brown. An overview of QEPCAD B: a tool for real quantifier elimination and formula simplification. *Journal of Japan Society for Symbolic and Algebraic Computation*, 10(1):13–22, 2003.

[10] G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas. Persistence barcodes for shapes. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometric Processing*, pages 124–135, New York, NY, USA, 2004. ACM Press.

[11] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. A singly-exponential stratification scheme for real semi-algebraic varieties and its applications. *Theoretical Computer Science*, 84:77–105, 1991.

[12] G. 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.

[13] H. Edelsbrunner. The union of balls and its dual shape. *Discrete Comput. Geom.*, 13(3-4):415–440, 1995.

[14] E. Fortuna, P. Gianni, P. Parenti, and C. Traverso. Algorithms to compute the topology of orientable real algebraic surfaces. *J. Symbolic Comput.*, 36(3-4):343–364, 2003.

[15] D. Halperin. Arrangements. In *Handbook of discrete and computational geometry*, CRC Press Ser. Discrete Math. Appl., pages 389–412. CRC, Boca Raton, FL, 1997.

[16] V. Koltun. Sharp bounds for vertical decompositions of linear arrangements in four dimensions. *Discrete Comput. Geom.*, 31(3):435–460, 2004.

Appendix A

To prove Proposition 1, we consider the following bi-graded double complex $\mathcal{M}^{p,q}$, with a total differential $D = \delta + (-1)^p d$, where

$$\mathcal{M}^{p,q} = \prod_{i_0,\ldots,i_p} C^q(A_{i_0,\ldots,i_p}).$$

$$\begin{tikzcd}
 & \vdots & \vdots & \vdots & \\
0 \arrow[r] & \prod_{i_0} C^3(A_{i_0}) \arrow[u, "d"] \arrow[r, "\delta"] & \prod_{i_0<i_1} C^3(A_{i_0,i_1}) \arrow[u, "d"] \arrow[r, "\delta"] & \prod_{i_0<i_1<i_2} C^3(A_{i_0,i_1,i_2}) \arrow[u, "d"] \arrow[r] & {} \\
0 \arrow[r] & \prod_{i_0} C^2(A_{i_0}) \arrow[u, "d"] \arrow[r, "\delta"] & \prod_{i_0<i_1} C^2(A_{i_0,i_1}) \arrow[u, "d"] \arrow[r, "\delta"] & \prod_{i_0<i_1<i_2} C^2(A_{i_0,i_1,i_2}) \arrow[u, "d"] \arrow[r] & {} \\
0 \arrow[r] & \prod_{i_0} C^1(A_{i_0}) \arrow[u, "d"] \arrow[r, "\delta"] & \prod_{i_0<i_1} C^1(A_{i_0,i_1}) \arrow[u, "d"] \arrow[r, "\delta"] & \prod_{i_0<i_1<i_2} C^1(A_{i_0,i_1,i_2}) \arrow[u, "d"] \arrow[r] & {} \\
0 \arrow[r] & \prod_{i_0} C^0(A_{i_0}) \arrow[u, "d"] \arrow[r, "\delta"] & \prod_{i_0<i_1} C^0(A_{i_0,i_1}) \arrow[u, "d"] \arrow[r, "\delta"] & \prod_{i_0<i_1<i_2} C^0(A_{i_0,i_1,i_2}) \arrow[u, "d"] \arrow[r] & {} \\
 & 0 \arrow[u, "d"] & 0 \arrow[u, "d"] & 0 \arrow[u, "d"] &
\end{tikzcd}$$

There are two spectral sequences (corresponding to taking horizontal or vertical filtrations respectively) associated with $\mathcal{M}^{p,q}$ both converging to $H^*_D(\mathcal{M})$. The first terms of these are $'E_1 = H_\delta \mathcal{M}, 'E_2 = H_d H_\delta \mathcal{M}$, and $''E_1 = H_d \mathcal{M}, ''E_2 = H_\delta H_d \mathcal{M}$. Because of the exactness of the generalized Mayer-Vietoris sequence, we have that,

$$'E_1 = \left|\begin{tikzcd}[row sep=small, column sep=small]
\vdots & \vdots & \vdots & \vdots & \vdots & \\
C^3(A) \arrow[u, "d"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & \cdots \\
C^2(A) \arrow[u, "d"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & \cdots \\
C^1(A) \arrow[u, "d"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & \cdots \\
C^0(A) \arrow[u, "d"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & 0 \arrow[u, "0"'] & \cdots
\end{tikzcd}\right.$$

and

$$'E_2 = \left|\begin{array}{llllll}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
H^3(A) & 0 & 0 & 0 & 0 & 0 \cdots \\
H^2(A) & 0 & 0 & 0 & 0 & 0 \cdots \\
H^1(A) & 0 & 0 & 0 & 0 & 0 \cdots \\
H^0(A) & 0 & 0 & 0 & 0 & 0 \cdots
\end{array}\right.$$

The degeneration of this sequence at E_2 shows that $H^*_D(\mathcal{M}) \cong H^*(A)$. The initial term $''E_1$ of the second spectral sequence is given by,

$$''E_1 = \left|\begin{tikzcd}[column sep=small]
\vdots & \vdots & \vdots & \\
\prod_{i_0} H^3(A_{i_0}) \arrow[r, "\delta"] & \prod_{i_0<i_1} H^3(A_{i_0,i_1}) \arrow[r, "\delta"] & \prod_{i_0<i_1<i_2} H^3(A_{i_0,i_1,i_2}) \arrow[r] & {} \\
\prod_{i_0} H^2(A_{i_0}) \arrow[r, "\delta"] & \prod_{i_0<i_1} H^2(A_{i_0,i_1}) \arrow[r, "\delta"] & \prod_{i_0<i_1<i_2} H^2(A_{i_0,i_1,i_2}) \arrow[r] & {} \\
\prod_{i_0} H^1(A_{i_0}) \arrow[r, "\delta"] & \prod_{i_0<i_1} H^1(A_{i_0,i_1}) \arrow[r, "\delta"] & \prod_{i_0<i_1<i_2} H^1(A_{i_0,i_1,i_2}) \arrow[r] & {} \\
\prod_{i_0} H^0(A_{i_0}) \arrow[r, "\delta"] & \prod_{i_0<i_1} H^0(A_{i_0,i_1}) \arrow[r, "\delta"] & \prod_{i_0<i_1<i_2} H^0(A_{i_0,i_1,i_2}) \arrow[r] & {}
\end{tikzcd}\right.$$

The cohomology groups $H^0(A_{i_0,\ldots,i_p})$ occurring as summands in the bottom row of $''E_1$ are isomorphic to the $\mathbb{Q}$-vector space of locally constant functions on $A_{i_0,\ldots,i_p}$ and the homomorphisms, $''d_1 : ''E_1^{p,0} \to ''E_1^{p+1,0}$ are then given by generalized restrictions, i.e. for

$$\phi \in \oplus_{1\le i_0<\cdots<i_p\le n} H^0(A_{i_0,\ldots,i_p}),$$

where each $\phi_{i_0,\ldots,i_p}$ is a locally constant function on $A_{i_0,\ldots,i_p}$,

$$''d_1(\phi)_{i_0,\ldots,i_{p+1}} = \sum_{i=0}^{p}(-1)^i \phi_{i_0,\ldots,\hat{i_i},\ldots,i_{p+1}}|_{A_{i_0,\ldots,i_{p+1}}}.$$

Proof (Proof of Proposition 1:). Since, $H^1(A_i) = 0$ for each i, clearly $''d_2^{0,1} = ''d_2^{1,0} = 0$. Thus, $''E_\infty^{1,0} = ''E_2^{1,0}$ and $''E_\infty^{0,1} = 0$. Thus, $H^1(A) \cong ''E_\infty^{1,0} \oplus ''E_\infty^{0,1} \cong ''E_2^{1,0}$.

A MAPLE Symbolic-Numeric Program for Solving the 2D-Eigenvalue Problem by a Self-consistent Basis Method

I.N. Belyaeva[1], N.A. Chekanov[1], A.A. Gusev[2], V.A. Rostovtsev[2], Yu.A. Ukolov[1], Y. Uwano[3], and S.I. Vinitsky[2]

[1] Belgorod State University, Studentcheskaja St.14, Belgorod, 308007, Russia
{chekanov, ukolov, ibelyaeva}@bsu.edu.ru
[2] Joint Institute for Nuclear Research, Dubna, Moscow Region 141980, Russia
vinitsky@thsun1.jinr.ru
[3] Future University-Hakodate, Hakodate, Japan

Abstract. The symbolic-numeric program SELFA for solving the the 2D boundary-value problem in self-consistent basis method is presented. The corresponding algorithm of this program using a conventional pseudocode is described too. As example, the energy spectrum and wave functions of E-type for generalized Henon–Heiles Hamiltonian were obtained.

1 Introduction

As is known, one of the more elaborated and widely applied methods of solving the eigenvalue problems describing the Hamiltonian systems is a diagonalization method [1]. However, in the case of multidimensional systems having a potential energy surface with few local minimuma[2], the efficiency of this method decreases in the energy region where in a classical limit a motion becomes chaotic [3]. An accuracy of numerical calculations of the corresponding set of energy levels of such type systems decreases drastically.

Usually one needs to diagonalize the Hamiltonian matrixes of a large dimension that leads to essential computer resource and run-time. In present paper the eigenvalue problem for a two-parametric generalized Henon–Heiles Hamiltonian corresponding to a non-integrable system is solved on the basis of a numerical method announced in [4]. In this method, the two-dimensional Schrödinger equation is reduced to a set of ordinary differential equations. Then the corresponding eigenvalue problem is solved directly instead of a rather cumbersome diagonalization of the above 2D problem. Such an approach is more promising due to an exact reduction by angular variable while numerical integration with a controlled accuracy is applied by a radial variable. This reduction is done with help of a self-consistent basis taking into account a discrete symmetry of the Hamiltonian under consideration that leads to a separation of the Hilbert space into the invariant subspaces and reduces the needed computer resources.

On the basis of this method we have developed the algorithm and MAPLE program SELFA for a symbolic-numeric solution of the two-dimensional Schrödinger equation. In this paper we give a unified description using a conventional pseudocode for the algorithm and program elaborated and present the energy spectrum and wave functions obtained by the program SELFA for a two-parametric generalized Henon–Heiles Hamiltonian.

2 Description of the Self-consistent Basis Method

We consider the stationary two-dimensional Schrödinger equation

$$\hat{H}(x,y)\psi(x,y) = E\psi(x,y), \tag{1}$$

with a two-parametric generalized Henon–Heiles Hamiltonian

$$\hat{H}(x,y) = -\frac{1}{2}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) + \frac{1}{2}(x^2+y^2) + b(x^2 y - \frac{1}{3}y^3) + c(x^2+y^2)^2, \tag{2}$$

where b and c are the real-valued parameters. In a polar coordinate system, $x = r\cos\varphi$, $y = r\sin\varphi$, Eqs. (1) and (2) take the form

$$\hat{H}(r,\varphi)\psi(r,\varphi) = E\psi(r,\varphi), \tag{3}$$

$$\hat{H}(r,\varphi) = -\frac{1}{2}\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \varphi^2}\right) + \frac{r^2}{2} + \frac{br^3}{2}\sin 3\varphi + cr^4. \tag{4}$$

A regular and bounded solution of the partial eigenvalue problem for Eqs. (3)–(4) can be found in terms of the Fourier series

$$\psi(r,\varphi) = \frac{A_0(r)}{2\sqrt{r}} + \frac{1}{\sqrt{r}}\sum_{l=1}^{n}[A_l(r)\cos l\varphi + B_l(r)\sin l\varphi]. \tag{5}$$

Using the Galerkin projection of Hamiltonian $\hat{H}(r,\varphi)$ and the unknown solution $u(r,\varphi)$ onto basis functions, $\sin l'\varphi$ and $\cos l'\varphi$ $(l' = 0,...,n)$,

$$\frac{1}{\pi}\int_0^{2\pi} \cos l'\varphi\left(\hat{H}(r,\varphi) - E\right)u(r,\varphi) = 0, \tag{6a}$$

$$\frac{1}{\pi}\int_0^{2\pi} \sin l'\varphi\left(\hat{H}(r,\varphi) - E\right)u(r,\varphi) = 0, \tag{6b}$$

we obtain the following infinite system of the differential equations of the second order:

$$\begin{array}{ll}
r^2 A_0'' + \alpha_0 A_0 - 2\beta B_3 = 0, & \\
r^2 A_1'' + \alpha_1 A_1 - \beta B_2 - \beta B_4 = 0, & \\
r^2 B_1'' + \alpha_1 B_1 - \beta A_2 + \beta A_4 = 0, & \\
r^2 A_2'' + \alpha_2 A_2 - \beta B_1 - \beta B_5 = 0, & \\
r^2 B_2'' + \alpha_2 B_2 - \beta A_1 + \beta A_5 = 0, & \\
r^2 A_3'' + \alpha_3 A_3 - \beta B_6 = 0, & \\
r^2 B_3'' + \alpha_3 B_3 - \beta A_0 + \beta A_6 = 0, & \\
r^2 A_l'' + \alpha_l A_l + \beta B_{l-3} - \beta B_{l+3} = 0, & 4 \le l \le n, \\
r^2 B_l'' + \alpha_l B_l - \beta A_{l-3} + \beta A_{l+3} = 0, & A_{k>n} = B_{k>n} = 0.
\end{array} \tag{7}$$

Here parameters α_l and β are defined by $\alpha_l = 2Er^2 - 2cr^6 - r^4 - l^2 + 1/4$, $\beta = br^5/3$. One can see that the system of equations (7) separates to four independent systems of the second-order ordinary linear differential equations (ODEs). This fact is a consequence of a discrete symmetry, C_{3V}, of the Henon–Heiles Hamiltonian (2) and corresponds to three irreducible representations:

$$\begin{array}{llllll}
\mathrm{A}_1: & A_{6l}, & B_{6l+3}, & l=0,1,..., & & \\
\mathrm{A}_2: & B_{6l}, & A_{6l-3}, & l=1,2,..., & & \\
\mathrm{E}_1: & A_{6l+1}, & B_{6l+2}, & B_{6l+4}, & A_{6l+5}, & l=0,1,..., \\
\mathrm{E}_2: & B_{6l+1}, & A_{6l+2}, & A_{6l+4}, & B_{6l+5}, & l=0,1,....
\end{array}$$

The E-type states of the type are double degeneracy because the eigenvalue problems for these two subsystems of the ODEs, (E_1 and E_2) have the same energy spectrum.

As an example, below we consider only E_2-type states. Using an appropriate transformation

$$\begin{array}{llll}
B_{6l+1} = z_{8l+1}, & B'_{6l+1} = z_{8l+2}, & A_{6l+2} = z_{8l+3}, & A'_{6l+2} = z_{8l+4}, \\
A_{6l+4} = z_{8l+5}, & A'_{6l+4} = z_{8l+6}, & B_{6l+5} = z_{8l+7}, & B'_{6l+5} = z_{8l+8},
\end{array}$$

of the above functions, $A_i(r)$, $B_j(r)$ $(i,j = 1,...,n)$, to the new ones, $z_k(r)$ $r = 1,...,2N$, where N is a number of equations of the ODEs of the second order, we rewrite the truncated set of linear second order ODEs (E_2) in the form of the linear first order ODEs

$$\begin{array}{ll}
{z_1}' - z_2 = 0, & {z_2}' + \alpha_1 z_1 - \beta(z_3 - z_5) = 0, \\
{z_3}' - z_4 = 0, & {z_4}' + \alpha_2 z_3 - \beta(z_1 - z_7) = 0, \\
{z_5}' - z_6 = 0, & {z_6}' + \alpha_4 z_5 + \beta(z_1 - z_9) = 0, \quad \dots .
\end{array} \tag{8}$$

To solve numerically the obtained eigenvalue problem one needs to reduce infinite interval $r \in (0,\infty)$ to a finite one $r \in [h, r_\infty]$, divide it by two subintervals, and construct the general solutions

$$\begin{array}{ll}
z_j^0(r) = \sum_{k=1}^{N} C_k z_{j,k}^0(r), & r \in [h, r_c], \\
z_j^\infty(r) = \sum_{k=1}^{N} C_{k+N} z_{j,k}^\infty(r), & r \in [r_c, r_\infty].
\end{array} \tag{9}$$

Here C_k are arbitrary coefficients and $z_{j,k}^0(r)$, $z_{j,k}^\infty(r)$, $j,k = 1,...,2N$ are independent basis solutions satisfying the set of equations (8) with boundary conditions,

$$z_{j,k}^0(h) = M_{j,k}^0, \quad z_{j,k}^\infty(r_\infty) = M_{j,k}^\infty, \tag{10}$$

where values $M_{j,k}^0$ and $M_{j,k}^\infty$ are determined from asymptotic expansions of regular solutions. Then, the continuity conditions

$$z_j^0(r_c) = z_j^\infty(r_c),$$

lead us to the algebraic eigenvalue problem

$$T_{jk} C_k = E_k C_k,$$

with a discrete spectrum E_k, i.e., at definite values of energy $E = E_s$ composing a low part of spectrum, $\sigma(E) = \{E_1, E_2, ..., E_N, ...\}$, of the eigenvalue problem (1)-(2).

3 Program Description

Following the description of the method for solving the eigenvalue problem (1)–(2), we present below the algorithm SELFA. The corresponding program SELFA has been implemented in a Maple Package.

Input:
N is the number of the second-order ordinary differential equations of the E_2 type;
$b,\ c$ are the real-valued parameters;
h and $Rend$ are boundary points;
r_c is a central point.
Output:
$\{E_s\}_{s=1}^N$ is a low part the energy spectrum;
$\{u_s(r,\phi)\}_{s=1}^N$ is the wave function corresponding to the energy value E_s in a form of the Fourier series.
The description of the local variables:
$u(r,\phi)$ is the local function in a form of the Fourier series.
$V(x,y)$ and $V(r,\phi)$ is the Henon–Heiles potential function in Cartesian and polar coordinates;
n is the number of the harmonics;
$A_k \equiv A_k(r),\ B_k \equiv B_k(r)\ (k = 0, 1, ..., n)$ are the coefficients of the Fourier series;
$Baseq(r,\phi)$ is the l.h.s. of the basic equation $(\hat{H} - E)u = 0$ in polar coordinates;
$BaseqA_l, BaseqB_l$ are the l.h.s. of the second-order ODEs (7);
$Etype$ is the set of the second-order ODEs (7);
$z_j(r)$ are the unknown functions of the system of ODEs of the first order (8);
ds are the l.h.s. of the first-order ODEs (8);
$dsys$ is the set of the first-order ODEs (8);
M^0 is the set of the initial conditions (10) on the left boundary for construction of sets of linear independent solutions of system of ODEs of the first order $dsys$;
$SOLN$ is the set of the linear independent solutions (9) of the set of the ODEs $dsys$;
M^∞ is the set of values of the linear independent solutions (9) at the right boundary point $Rend$;
T^0, T^∞ are the sets of values of the linear independent solutions (9) at the central point r_c;
T is the matrix of the continuity conditions;
$\{C_j\}$ and $\{C_{j;s}\}$ are auxiliary eigenvectors;

```
1:  $n := N + N/2 - 1;$
2:  $u(r,\phi) := \dfrac{A_0}{2\sqrt{r}} + \dfrac{1}{\sqrt{r}}\sum_{k=1}^{n}( A_k\cos(k\,\phi) + B_k\sin(k\,\phi));$
3:  $V(x,y) := \frac{1}{2}x^2 + \frac{1}{2}y^2 + b\,(x^2\,y - \frac{1}{3}y^3) + c\,(x^2+y^2)^2;$
    $V(r,\phi) := subs(x \to r\cos(\phi), y \to r\sin(\phi), V(x,y));$
4:  $Baseq(r,\phi) := \left(\dfrac{1}{r^2}\dfrac{\partial^2}{\partial\phi^2} + 2(E + \dfrac{1}{8r^2} - V(r,\phi))\right)u(r,\phi);$
    for $l$ from 0 to $n$ do
        $BaseqA_l := r^2\mathrm{coeff}(Baseq(r,\phi), \cos(l\phi));$
        if $l \geq 1$ then
            $BaseqB_l := r^2\mathrm{coeff}(Baseq(r,\phi), \sin(l\phi));$
        end if;
    end do;
5:  for $i$ from 0 to $(n+1)/6 - 1$ do
        $Etype_{4i} := BaseqB_{6i+1};$
        $Etype_{4i+1} := BaseqA_{6i+2};$
        $Etype_{4i+2} := BaseqA_{6i+5};$
        $Etype_{4i+3} := BaseqB_{6i+4};$
    end do;
    for $i$ from 0 to $N-1$ do
    for $j$ from 0 to $(n+1)/6 - 1$ do
        $Etype_i := subs(\{B_{6j+1} \to z_{8j+1}(r), A_{6j+2} \to z_{8j+3}(r),$
            $A_{6j+4} \to z_{8j+5}(r), B_{6j+5} \to z_{8j+7}(r)\}, Etype_i);$
    end do;
    end do;
6:  for $i$ from 0 to $N-1$ do
        $ds_{2i+1} := \dfrac{dz_{2i+1}(r)}{dr} - z_{2i+2}(r) = 0;$
        $ds_{2i+2} := \dfrac{dz_{2i+2}(r)}{dr} + \dfrac{Etype_i}{r^2} = 0;$
    end do;
    $dsys := \{ds_1, ds_2, ..., ds_{2N}\};$
7:  for $i$ from 1 to $N$ do
        $SOLN := dsolve(\{dsys, z_j(h) = M^0_{i,j}, (j = 1,...,2N)\}, \{z_j(r)\}_{j=1}^{2N});$
        $T^0_{ij} := SOLN : z_j(r_c), \quad (j = 1,...,2N);$
        $SOLN := dsolve(\{dsys, z_j(Rend) = M^\infty_{i,j}, (j = 1,...,2N)\}, \{z_j(r)\}_{j=1}^{2N});$
        $T^\infty_{ij} := SOLN : z_j(r_c), \quad (j = 1,...,2N);$
    end do;
8:  for $i$ from 1 to $N$ do
    for $j$ from 1 to $2N$ do
        $T_{ij} := T^\infty_{ij} - T^0_{ij};$
    end do;
    end do;
9:  $T_{ij}C_j = EC_j \quad \to \quad \{E_s, \{C_{j;s}\}\}_{s=1}^{N}$
10: for $s$ from 1 to $N$ do
```

```
        $SOLN_s := dsolve(\{dsys, z_j(h) = C_{j;s}, (j = 1, ..., 2N)\}, \{z_j(r)\}_{j=1}^{2N});$
        $u_s(r,\phi) := SOLN_s :$
            $\sum_{j=0}^{(n+1)/6-1}\Big(z_{8j+3}(r)\cos((6j+2)\phi) + z_{8j+5}(r)\cos((6j+4)\phi)$
                $+z_{8j+1}(r)\sin((6j+1)\phi) + z_{8j+7}(r)\sin((6j+5)\phi)\Big);$
    end do;
```

Remark: This program involves the following sequence of the steps.
Steps 1–2. The wave function in the form of the Fourier series is presented.
Steps 3–4. Construction of the set of the second-order ODEs. At step 4 instead of formula (6) the standard MAPLE procedure "coeff" for extracting coefficients affecting $\cos(l\phi)$ and $\sin(l\phi)$ is used.
Steps 5–6. Construction of the set of the first-order ODEs.
Step 7. Construction of the linear independent solutions with help of the conventional subroutine *dsolve* of a Maple package for numerical solving of a set of the $2N$ first order ODEs.
Step 8. Construction of continuity conditions matrix.
Step 9. Evaluation of the energy spectrum.
Step 10. Evaluation of the eigenfunctions.

4 Examples of SELFA Program Runs

The eigenvalues and functions of E_2-types were calculated by means of the program SELFA for a generalized Henon–Heiles Hamiltonian. Values of the lowest energy levels together with the ones obtained by the diagonalization method [5] are presented in Table 1. The energy spectrum in Ref. [5] was obtained by a direct diagonalization of the Hamiltonian 495×495 matrix but in our approach the same accuracy was achieved by solving system (8) of $2N = 16$ differential equations of the first order. It is shown that in our approach one needs a less computer resource and running time in comparison with the diagonalization method. The program SELFA was also used to calculate the corresponding wave functions, two of which are shown in Fig. 1. One can see that a symmetric struc-

Table 1. The energy spectrum of E-type for the Hamiltonian (2) at fixed values of parameters $b = 0.04416$, $c = 0.00015$

s	$E_{diag}[4]$	E	s	$E_{diag}[4]$	E
1.	1.999384	1.999372	7.	6.005955	6.005972
2.	2.999628	2.999641	8.	6.976317	6.976625
3.	3.992368	3.992439	9.	6.988910	6.989034
4.	4.990280	4.990394	10.	7.964477	7.964810
5.	5.002921	5.002935	11.	7.989611	7.989745
6.	5.980721	5.980968	12.	8.014769	8.014855

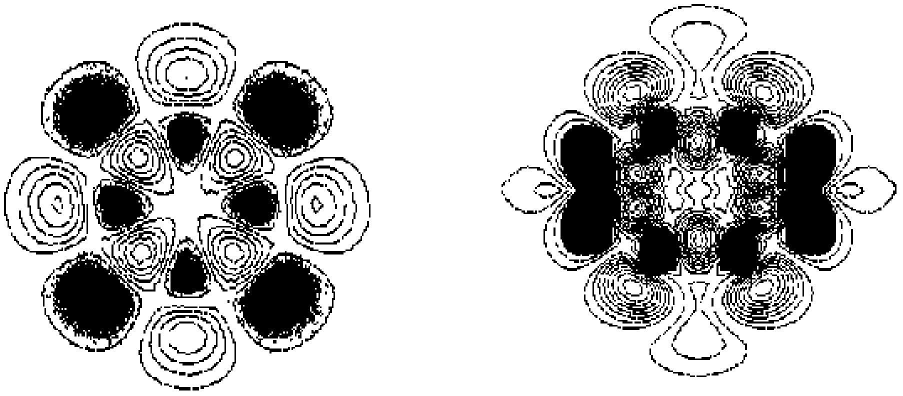


Fig. 1. Isolines of the E$_2$-type wave functions $u_9(x,y)$ (in left panel) and $u_{11}(x,y)$ (in right panel) of the generalized Henon–Heiles Hamiltonian (2) at $b = 4.416 \cdot 10^{-2}$ and $c = 1.5 \cdot 10^{-4}$ (dark and white domains correspond to negative and positive values, respectively)

ture of isolines of the wave functions $u_9(x,y)$ and $u_{11}(x,y)$ reveals explicitly the C_{3V} symmetry of the generalized Henon–Heiles Hamiltonian (2).

5 Conclusions

A MAPLE program SELFA for a symbolic-numeric solution of the two-dimensional Schrödinger equation in self-consistent basis method is presented. An efficiency of this program is shown on an example of the generalized Henon–Heiles Hamiltonian (2) for which the lowest energy levels and wave functions were calculated and a comparison was made with the results obtained by diagonalization method. The program SELFA may further be applied for studying the eigenproblem for different Hamilton operators and, for example, for investigating the avoiding crossing phenomena of eigenenergies, etc.

One of topical tasks here is a comparison of numeric and analytic results for spectrum and wave functions in a vicinity of avoiding crossing of energy levels with respect to parameters that can be performed with help of the above algorithm SELFA and programs of normalization and quantization of the polynomial Hamiltonians [6,7]. Such a comparison allows one to reveal the nature of quantum chaos of the Hamiltonian systems that in quantum case has quantum counterparts like a degeneracy of the energy levels and tunnelling through a potential barrier with few local extrema and to determine various decay mechanisms of the quantum system under consideration. A study in the field will be a subject of our further investigations.

Acknowledgment

Authors express their gratitude to RFBR for support by grant No. 03-02-16263, by grant No. BKG 003-04 from BSU, and by Grant-in-Aid Scientific Research No. 16560050 from JSPS.

References

1. Wilkinson, J.H., Reinsch, C.: Handbook for Automatic Computation, Vol. 2: Linear Algebra. Springer-Verlag, New York (1971)
2. Gutzwiller, M.C.: Chaos in Classical and Quantum Mechanics. Springer, New York (1990)
3. Bolotin, Yu. L., Gonchar, V.Yu., Tarasov, V.N., Chekanov, N.A.: Phys. Lett. A **135** (1989), 29
4. Vinitsky, S.I., Pak, D.N., Rostovtsev, V.A., Chekanov, N.A., Ukolov, Yu.A.: In the collection of theses of reports of 54th International Meeting on Nuclear Spectroscopy. Belgorod, 22–25 June, 2004, p. 321
5. Bolotin, Yu.L., Vinitsky, S.I., Gonchar, V.Yu. et al.: JINR preprint, P4-89-590, Dubna (1989) 26
6. Gusev, A.A., Chekanov, N.A., Rostovtsev, V.A., Uwano, Y., Vinitsky, S.I.: In: Computer Algebra in Scientific Computing, V.G. Ganzha, E.W. Mayr, E.V. Vorozhtsov (Eds.), Technische Universitat, Munchen, 147 (2002)
7. Ukolov, Yu.A., Chekanov, N.A., Gusev, A.A., Rostovtsev, V.A., Vinitsky, S.I. and Uwano, Y.: Computer Physics Communications **166** (2005), 66

RelView – An OBDD-Based Computer Algebra System for Relations

Rudolf Berghammer and Frank Neumann

Institut für Informatik und Praktische Mathematik,
Christian-Albrechts-Universität Kiel,
Olshausenstraße 40, D-24098 Kiel

Abstract. We present an OBDD-based Computer Algebra system for relational algebra, called RelView. After a short introduction to the OBDD-implementation of relations and the system, we exhibit its application by presenting two typical examples.

1 Introduction

Since many years relational algebra has widely been used by mathematicians and computer scientists as a convenient means for problem solving; see e.g., [12,3]. One main reason for the use of relational algebra is that it has a fixed and surprisingly small set of operations all of which can efficiently and with reasonable effort be implemented on finite carrier sets using, e.g., Boolean arrays, linked lists, or ordered binary decision diagrams (OBDDs). Thus, a Computer Algebra system for relational algebra can be implemented with reasonable effort, too. In this paper, we want to give an impression of such a system, called RelView, which has been developed at Kiel University since 1993 and is available free of charge, see http://www.informatik.uni-kiel.de/~progsys/relview.html.

RelView can be used to solve many different tasks while working with relational algebra, relation-based discrete structures, and relational programs. E.g., it can support relation-algebraic reasoning. In this field typical applications are the search for counter-examples or the detection of new properties. For these activities, "playing" and "experimenting" with relation-algebraic expressions is essential and this is one of the purpose RelView has been designed for. In particular, the interactive nature of the system allows to add, change and remove relations and their representations, and makes it possible to invoke computations at every time within a working session. A further application domain is programming. This not only concerns the implementation of relational algorithms using the system's programming language, but also typical tasks appearing in (formal) program development like specification testing, detection of loop invariants and other important properties necessary for correctness proofs, rapid prototyping, and improving efficiency. Third, the advantages of the system when using it in teaching and for visualization respectively animation should be mentioned. We have recognized that visualizing advanced concepts of relational algebra (like relational domain constructions) in RelView is very helpful. Furthermore, since

RelView allows computations not only to be executed fully-automatically but also in a stepwise fashion, it is a very good means to demonstrate how a certain algorithm works. Finally, it should be mentioned that we found it very attractive to use RelView for producing good examples in teaching, which frequently have been proven to be the key of fully understanding a concept. Of course, we are not able to show all these aspects of RelView in this paper.

2 Relation-Algebraic Preliminaries

We write $R : X \leftrightarrow Y$ if R is a relation with domain X and range Y, i.e., a subset of $X \times Y$. If the sets X and Y of R's *type* $X \leftrightarrow Y$ are finite, we may consider R as a Boolean matrix. Since this Boolean matrix interpretation is well suited for many purposes and also used as one possibility of RelView to depict relations, in the following we often use matrix terminology and matrix notation. Especially, we speak about the rows and columns of R and write R_{xy} instead of $(x, y) \in R$. We assume the fundamentals of relational algebra to be known, viz. R^{T} (*transposition*), $\overline{R}$ (*complement*), $R \cup S$ (*union*), $R \cap S$ (*intersection*), $R\,S$ (*composition*), $R \subseteq S$ (*inclusion*), and the special relations O (*empty relation*), L (*universal relation*), and I (*identity relation*). A relation R is *univalent* if $R^{\mathsf{T}}R \subseteq \mathsf{I}$, *total* if $R\mathsf{L} = \mathsf{L}$, *injective* if R^{T} is univalent, *reflexive* if $\mathsf{I} \subseteq R$, *antisymmetric* if $R \cap R^{\mathsf{T}} \subseteq \mathsf{I}$, *transitive* if $RR \subseteq R$, *symmetric* if $R = R^{\mathsf{T}}$, and *irreflexive* if $R \subseteq \overline{\mathsf{I}}$. A univalent and total relation is a *mapping*, a reflexive, antisymmetric, and transitive relation is a *partial order*, and a reflexive and transitive relation is a *quasi order*.

By $\mathrm{syq}(R, S) := \overline{R^{\mathsf{T}}\overline{S}} \cap \overline{\overline{R}^{\mathsf{T}}S}$ the *symmetric quotient* $\mathrm{syq}(R, S) : Y \leftrightarrow Z$ of two relations $R : X \leftrightarrow Y$ and $S : X \leftrightarrow Z$ is defined. Many properties of this construct can be found in [12]. In this paper we only need the equivalence

$$\mathrm{syq}(R, S)_{yz} \iff \forall\, x : R_{xy} \leftrightarrow S_{xz}\,. \qquad (1)$$

Relational algebra provides some possibilities for modeling sets. The first modeling which we will apply in this paper uses *vectors*. These are relations v with $v = v\mathsf{L}$. For v being of type $X \leftrightarrow Y$ this condition means: Whatever set Z and universal relation $\mathsf{L} : Y \leftrightarrow Z$ we choose, an element $x \in X$ is in relationship $(v\mathsf{L})_{xz}$ either to none element $z \in Z$ or to every element $z \in Z$. Since for a vector the range is irrelevant, in the following we consider mostly vectors $v : X \leftrightarrow \mathbf{1}$ with a specific singleton set $\mathbf{1} = \{\perp\}$ as range and omit in such cases the second subscript, i.e., write v_x instead of $v_{x\perp}$. Such a vector can be considered as a Boolean matrix with exactly one column, i.e., as a Boolean column vector, and *represents* the subset $\{x \in X \mid v_x\}$ of its domain X.

We will also use injective mappings for modeling subsets. Given an injective mapping $\imath : Y \leftrightarrow X$, we may consider Y as a subset of X by identifying it with its image under $\imath$. If Y is actually a subset of X and $\imath$ is the identity mapping from Y to X, then the vector $\imath^{\mathsf{T}}\mathsf{L} : X \leftrightarrow \mathbf{1}$ represents Y as subset of X in the sense above. Clearly, also the transition in the other direction is possible, i.e., the generation of an injective mapping

$$\mathrm{inj}(v) : Y \leftrightarrow X \qquad \mathrm{inj}(v)_{yx} :\Longleftrightarrow y = x \tag{2}$$

from a given vector $v : X \leftrightarrow \mathbf{1}$ representing the subset Y of X. We call $\mathrm{inj}(v)$ the *injective mapping generated* by the vector v.

As a third possibility to model subsets of a given set X we will use the set-theoretic *membership relation* defined by (2^X is the power set of X)

$$\mathsf{M} : X \leftrightarrow 2^X \qquad \mathsf{M}_{xY} :\Longleftrightarrow x \in Y\,. \tag{3}$$

Using matrix terminology, a combination of injective mappings and membership relations leads to a *column-wise representation* of sets of subsets. More specifically, if the vector $v : 2^X \leftrightarrow \mathbf{1}$ represents a subset $\mathfrak{S}$ of 2^X in the sense above, then for all $x \in X$ and $Y \in \mathfrak{S}$ we get the equivalence of $x \in Y$ and $(\mathsf{M}\,\mathrm{inj}(v)^{\mathsf{T}})_{xY}$ due to (2) and (3). This means that the elements of $\mathfrak{S}$ are represented precisely by the columns of the relation $C := \mathsf{M}\,\mathrm{inj}(v)^{\mathsf{T}} : X \leftrightarrow \mathfrak{S}$. A further consequence of this fact is that $\overline{C^{\mathsf{T}}\overline{C}} : \mathfrak{S} \leftrightarrow \mathfrak{S}$ is the relation-algebraic specification of set inclusion on $\mathfrak{S}$, that is for all $Y, Z \in \mathfrak{S}$ we have that $\overline{(C^{\mathsf{T}}\overline{C})}_{YZ}$ iff $Y \subseteq Z$.

Using some well-known correspondences between certain logical constructions and relation-algebraic operations (see e.g., [12]) it can easily be shown that if $R : X \leftrightarrow X$ is a quasi order and $v : X \leftrightarrow \mathbf{1}$ represents a subset Y of X, then the *set of greatest elements* of Y with respect to R is represented by the vector

$$max(R, v) := v \cap \overline{\overline{R}^{\mathsf{T}} v} : X \leftrightarrow \mathbf{1}\,. \tag{4}$$

In Section 5 we will apply (4) to compute maximum cliques. This means that X is the power set 2^V of the set V of vertices of the input graph g, the first argument of the relational function *max* of (4) is the *size-comparison relation*

$$\mathsf{S} : 2^V \leftrightarrow 2^V \qquad \mathsf{S}_{AB} :\Longleftrightarrow |A| \le |B|\,, \tag{5}$$

and the second argument of *max* is a vector representing the set of cliques of g.

3 Implementation of Relations Using OBDDs

Assuming the reader to be familiar with the basic facts of OBDDs (as e.g., presented in [5,13]), we sketch in the following how to implement relations with their help. For a complete description we refer to the Ph.D. theses [10,11].

OBDDs are an efficient data structure to implement very large Boolean functions. Our implementation of relation uses this fact. We will illustrate it by a small example. Assume sets $X := \{a, b, c, d\}$ and $Y := \{r, s\}$ and the relation

$$R : X \leftrightarrow Y \qquad R := \{\,(a, r), (c, r), (c, s)\,\}.$$

By means of the canonical binary encodings $c_X : X \to \mathbb{B}^2$ and $c_Y : Y \to \mathbb{B}$ of X and Y, specified by $c_X(a) = 0, 0, c_X(b) = 0, 1, c_X(c) = 1, 0, c_X(d) = 1, 1$ respectively $c_Y(r) = 0, c_Y(s) = 1$, we can define a Boolean function $f_R : \mathbb{B}^3 \to \mathbb{B}$

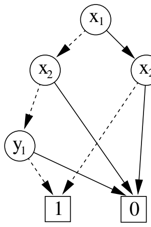


Fig. 1. OBDD for the function f_R

such that $f_R(x_1, x_2, y_1) = 1$ iff $c_X^{-1}(x_1, x_2)$ and $c_Y^{-1}(y_1)$ are defined and related via R. Then each satisfying assignment of the function f_R corresponds to a pair of the relation R and we obtain f_R in disjunctive normal form as

$$f_R(x_1, x_2, y_1) = (\overline{x_1} \wedge \overline{x_2} \wedge \overline{y_1}) \vee (x_1 \wedge \overline{x_2} \wedge \overline{y_1}) \vee (x_1 \wedge \overline{x_2} \wedge y_1)\,.$$

Here, for instance, the first clause $\overline{x_1} \wedge \overline{x_2} \wedge \overline{y_1}$ expresses the fact that the two elements $c_X^{-1}(0, 0) = a$ and $c_Y^{-1}(0) = r$ are related via the relation R. If we use the fixed variable ordering $x_1 < x_2 < y_1$, then we get the OBDD for the function f_R shown in Figure 1, where a continuous edge means that the value of its source is 1 and a dotted edge means that this value is 0.

The implementation of relations in the shown way leads to the problem that one OBDD can implement several relations. For instance, it is easy to check that the Boolean function obtained from the relation

$$S : Y \leftrightarrow X \qquad\qquad S := \{\, (r, a), (s, a), (s, b) \,\}$$

coincides with f_R. This means that R and S are implemented by the same OBDD. However, the problem can be solved by additionally storing the sizes of the carrier sets since R is the only relation of type $X \leftrightarrow Y$ which leads to f_R.

As general technique, using the two canonical binary encodings $c_X : X \rightarrow \mathbb{B}^m$ and $c_Y : Y \rightarrow \mathbb{B}^n$ (where $m = \lceil \log |X| \rceil$ and $n = \lceil \log |Y| \rceil$) a relation $R : X \leftrightarrow Y$ is implemented by the two sizes $|X|$ and $|Y|$ and the OBDD of the Boolean function $f_R : \mathbb{B}^{m+n} \rightarrow \mathbb{B}$, such that $f_R(x_1, \ldots, x_m, y_1, \ldots, y_n) = 1$ iff the two decodings $c_X^{-1}(x_1, \ldots, x_m)$ and $c_Y^{-1}(y_1, \ldots, y_n)$ are defined and related via the relation R and the variable ordering is $x_1 < \ldots < x_m < y_1 < \ldots < y_n$.

Based on this implementation of relations, in the course of the Ph.D. theses [10,11] many relation-algebraic operations (including, of course, those of Section 2) have been implemented as operations on OBDDs. Due to lack of space we can not go into detail. However, it should be emphasized that the specific variable ordering with the variables encoding the elements of the domain followed by those encoding the elements of the range allows a very efficient implementation of membership relations and size-comparison relations.

In the case of the membership relation $\mathsf{M} : X \leftrightarrow 2^X$ of (3) we obtain a Boolean function $f_{\mathsf{M}} : \mathbb{B}^{m+n} \rightarrow \mathbb{B}$ with $m \leq \log |X| + 1$ variables for encoding the elements of the domain X and $n = |X|$ variables for encoding the elements of the range 2^X. It is defined by

$$f_{\mathsf{M}}(x_1,\dots x_m,y_1,\dots,y_n) = \begin{cases} y_{c+1} & : \quad c \leq n-1 \\ 0 & : \quad \text{otherwise,} \end{cases}$$

where $c := c_X^{-1}(x_1,\dots,x_m)$ is the decoding of the bitstring $x_1,\dots,x_m$.

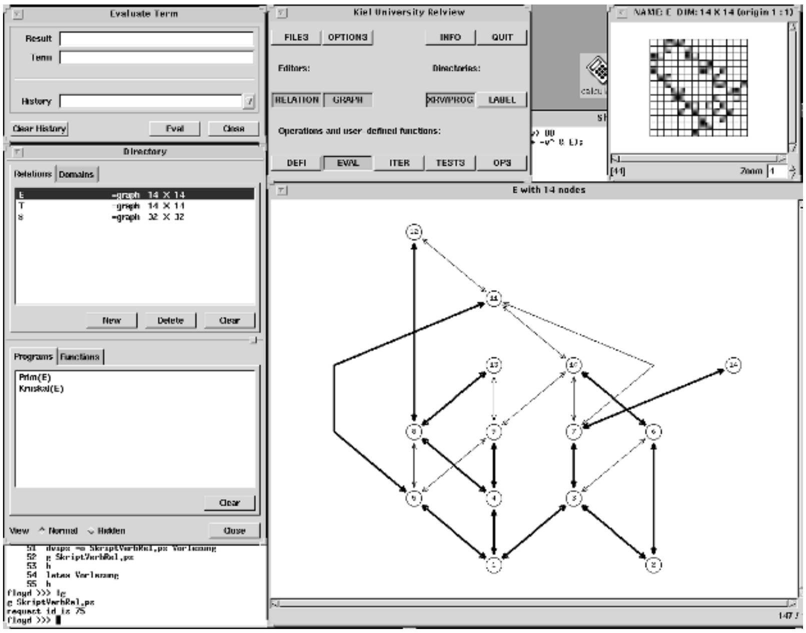
Given a variable ordering such that the variables $x_1,\dots,x_m$ are tested before the variables $y_1,\dots,y_n$, the number of nodes of the OBDD of f_{M} with respect to this variable ordering is bounded by $2^m - 1 + n + 2$, that is, by $3|X|+1$. The argument is that the x-variables are tested in a binary decision tree with at most $2^m - 1$ inner nodes and at most 2^m leaves. (The case that $|X|$ is a power of 2 leads to a complete binary decision tree with $2^m - 1$ inner nodes and 2^m leaves.) Each leaf is replaced by a test of the corresponding y-variables if the number of the binary representation is not greater than $n-1$. All other leaves are deleted and the incoming edges are directed to the sink 0. There may be reduction rules applicable to this OBDD but this only reduces the number of nodes. The result follows by adding 2 nodes for the sinks.

In the case that $|X|$ is a power of 2, the Boolean function f_{M} of the membership relation $\mathsf{M} : X \leftrightarrow 2^X$ is the well-known *direct storage access function* DSA_n (see e.g., [13]). Here m x-variables address $n = 2^m$ y-variables which decide about the output. For DSA_n the fraction of variable orderings leading to a non-polynomial OBDD size converges to 1, and this also holds for the function f_{M}. A lot of RelView programs work with membership relations, especially if one deals with hard problems or uses the system for supporting the engineering and validation of relational specifications. Therefore, the choice of our specific variable ordering is justified. The large number of bad variable orderings also suggests not to change the variable ordering during the computation process when working with this relation. This has been confirmed by experimental studies showing that the use of different variable orderings and reordering techniques leads in most cases to a much higher computation time in comparison to a computation with our fixed variable ordering.

For the size-comparision relation $\mathsf{S} : 2^X \leftrightarrow 2^X$ of (5) an OBDD-implementation has been developed in [11] which exactly uses $2 + |X|(|X|+1)$ OBDD-nodes for the proposed variable ordering.

4 The Computer Algebra System RelView

RelView (see [1,2]) is a totally interactive and completely graphic-oriented "specific purpose" Computer Algebra system for dealing with relational algebra. In it all data are represented as (of course, finite) relations. Especially when applied for prototyping or to solve hard problems via enumeration, the system often works on very large objects since, e.g., a membership relation appears during a computation. Therefore, it uses a very efficient internal implementation of relations via OBDDs following the approach sketched in Section 3. Externally, relations are visualized in two different ways; see the snapshot of Figure 2. For relations of type $X \leftrightarrow X$ RelView offers a representation as directed graphs, including sophisticated algorithms for drawing them nicely. Alternatively, arbitrary relations may be depicted as Boolean matrices. This second representation



```
E                 =graph  14 X 14
T                 =graph  14 X 14
S                 =graph  32 X 32
```

```
Prim(E)
Kruskal(E)
```

```
    51  dvips -o SkriptVerbRel.ps Vorlesung
    52  g SkriptVerbRel.ps
    53  h
    54  latex Vorlesung
    55  h
floyd >>> lg
g SkriptVerbRel.ps
request id is 75
floyd >>> █
```

Fig. 2. The screen of RelView

is very useful for visually editing and also for discovering various structural properties that are not evident from a representation of relations as directed graphs. Although the detailed appearance of the windows of RelView depends on specific conceptions of the users, typically the main windows (viz. evaluation window, menu window, directory window, window of the relation editor, window of the graph editor) look as in the screen snapshot of Figure 2.

The main purpose of RelView is the evaluation of relation-algebraic expressions which are constructed from the relations of the workspace using many predefined operations (like `-`, `^`, `&`, `|`, and `*` for complement, transposition, intersection, union, and composition) and tests (like `empty` for testing emptiness) on them, user-defined relational functions, and user-defined relational programs. A relational function is of the form F`(`$X_1, \ldots, X_n$`) =` t, where F is the function name, the X_i, $1 \leq i \leq n$, are the formal parameters (standing for relations), and t is a relation-algebraic expression over the relations of the workspace that can additionally contain the formal parameters X_i. A relational program essentially is a while-program based on the datatype of relations. It starts with a head line containing the program's name and a list of formal parameters. Then the declaration part follows. The third part is its body, a sequence of statements which are separated by semicolons and terminated by the return-clause.

5 A Graph-Theoretic Application: Maximum Cliques

Throughout this section, we fix an undirected graph $g = (V, E)$ and assume that it is represented by the symmetric and irreflexive *adjacency relation* $R : V \leftrightarrow V$.

That is, we suppose for all $x, y \in V$ that R_{xy} iff $\{x, y\}$ is an edge from E. A *clique* of g is a subset C of V in which every pair x, y of distinct vertices is connected by an edge, i.e., $\{x, y\} \in E$. The clique C is called *maximum* if its cardinality is maximum. The maximum clique problem is NP-hard. Moreover, it is even one of the hardest problems with respect to polynomial-time approximability.

We want to compute maximum cliques of g using RelView. First we develop a vector of type $2^V \leftrightarrow \mathbf{1}$, which represents the set of all cliques of g. Let X be an arbitrary set of vertices. We decide whether X is a clique, using the predicate-logic specification of X to be a clique, the definition (3) of the membership relation $\mathsf{M} : X \leftrightarrow 2^X$, and some simple laws of predicate logic respectively well-known correspondences between logical and relation-algebraic constructions:

$$\begin{aligned}
X \text{ is a clique of } g &\iff \forall\, x, y : x \in X \wedge y \in X \wedge y \neq x \rightarrow R_{yx} \\
&\iff \forall\, x : x \in X \rightarrow (\forall\, y : y \in X \rightarrow (y = x \vee R_{yx})) \\
&\iff \forall\, x : \mathsf{M}_{xX} \rightarrow (\forall\, y : \mathsf{M}_{yX} \rightarrow (\mathsf{I} \cup R)_{yx}) \\
&\iff \forall\, x : \mathsf{M}_{xX} \rightarrow (\neg \exists\, y : \mathsf{M}_{yX} \wedge \overline{\mathsf{I} \cup R}_{yx}) \\
&\iff \forall\, x : \mathsf{M}_{xX} \rightarrow \neg(\mathsf{M}^{\mathsf{T}}\overline{\mathsf{I} \cup R})_{Xx} \\
&\iff \neg \exists\, x : \mathsf{M}^{\mathsf{T}}_{Xx} \wedge (\mathsf{M}^{\mathsf{T}}\overline{\mathsf{I} \cup R})_{Xx} \\
&\iff \neg \exists\, x : (\mathsf{M}^{\mathsf{T}} \cap \mathsf{M}^{\mathsf{T}}\overline{\mathsf{I} \cup R})_{Xx} \wedge \mathsf{L}_x \qquad\qquad \mathsf{L} : V \leftrightarrow \mathbf{1} \\
&\iff \overline{(\mathsf{M}^{\mathsf{T}} \cap \mathsf{M}^{\mathsf{T}}\overline{\mathsf{I} \cup R})\mathsf{L}}_X
\end{aligned}$$

Next, we remove the subscript X from the last expression of the derivation following the vector-representation of sets introduced in Section 2. To improve efficiency, after that we apply symmetry of R in combination with some well-known relation-algebraic laws to transpose only a "row vector" instead of a relation of type $V \leftrightarrow 2^V$, yielding

$$\mathit{cliques}(R) := \overline{\mathsf{L}^{\mathsf{T}}(\mathsf{M} \cap \overline{\mathsf{I} \cup R}\mathsf{M})}^{\mathsf{T}} : 2^V \leftrightarrow \mathbf{1} \tag{6}$$

as relation-algebraic specification of the vector representing the set of all cliques of g. (Using an OBDD-implementation of relations, transposition of a relation with domain or range $\mathbf{1}$ is trivial. It only means to exchange domain and range, the OBDD remains unchanged. See [11] for details.)

Now, let $\mathsf{S} : 2^V \leftrightarrow 2^V$ be the size-comparison relation on 2^V as introduced by (5). Then an application of the relational function *max* of (4) to S and the vector *cliques*(R) immediately yields the following vector-representation of the set $\mathfrak{M}$ of all maximum cliques of g:

$$\mathit{maxcliques}(R) := \mathit{max}(\mathsf{S}, \mathit{cliques}(R)) : 2^V \leftrightarrow \mathbf{1}\,. \tag{7}$$

The column-wise representation of $\mathfrak{M}$ by $\mathsf{M}\,\mathrm{inj}(\mathit{maxcliques}(R))^{\mathsf{T}} : V \leftrightarrow \mathfrak{M}$ now is a direct consequence of the technique explained in Section 2.

Each of the above relation-algebraic specifications can be easily translated into the programming language of RelView. Especially, (4), (6), and (7) read as RelView-code as follows:

```
max(R,v) = -v & -(-R^ * v).
cliques(R)
  DECL M
  BEG  M = epsi(O(R))
       RETURN -(L1n(R) * (M & -refl(R) * M))^
  END.
maxcliques(R) = max(cardrel(O(R)),cliques(R)).
```

The RelView tool automatically allows to generate uniform random relations, also of a specific kind and density; see [11] for details. We have applied this feature to test the efficiency of our approach, where we used a Sun-Fire 880 workstation running Solaris 9 at 750 MHz. Figure 3 shows some experimental results for randomly generated symmetric and irreflexive adjacency relations. The number of vertices N of the random graph is listed at the x-axis and the time needed to execute `maxcliques` is listed at the y-axis. The four curves have been obtained by varying N from 50 to 500 by steps of 50 vertices and the probability of a pair $\{x, y\}$ to be an edge of a graph from 10% to 25% by steps of 5%. We performed at least 20 experiments for each N and each density and computed in all cases the arithmetic mean of the execution times.

Because of lack of memory, in the case of 25% density we have not been able to deal with $N > 300$ vertices. If, however, we restricted us to more sparse graphs, larger numbers of vertices could be treated successfully. For example, $N = 700$ and a density of 10% led to approximately 12 minutes execution time and, for the same density, $N = 1000$ led to roughly 100 minutes.

Using the same environment, we have also tested `maxcliques` on some DIMACS benchmaks (see [7]) with up to 500 vertices. The results are shown in Table 1, where the fourth column contains the sizes of the maximum cliques and the sixth column shows the computation times in seconds. Apart from the size of a maximum clique and the computation time it is very interesting to see the number of maximum cliques for the considered benchmarks. For instance, in the case of MANN_a9 there are exactly 9540 maximum cliques.

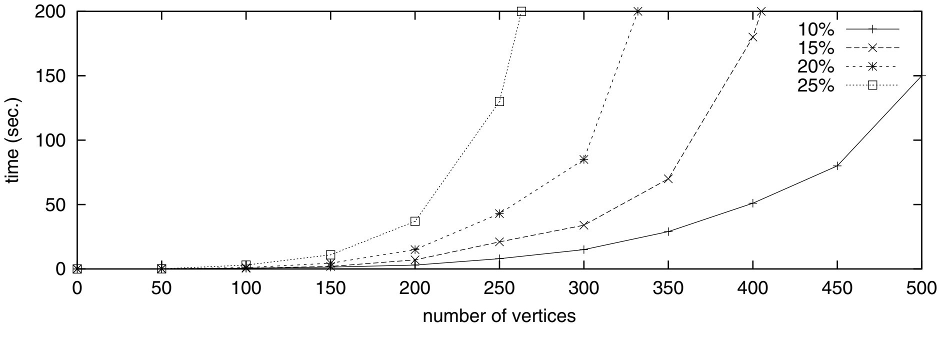


Fig. 3. Computational results on randomly generated instances

Table 1. Computational results on DIMACS Benchmarks

Problem	Vertices	Edges	Clique Size	# Max. Cliques	Time
c-fat200-1	200	1534	12	14	0.77
c-fat200-2	200	3235	24	1	0.68
c-fat500-1	500	4459	14	19	8.96
c-fat500-2	500	9139	26	19	7.92
johnson8-2-4	28	210	4	105	0.02
hamming6-4	64	704	4	240	0.81
MANN_a9	45	918	16	9540	0.30

Of course, in view of efficiency our approach cannot compete with special programs and tools for the *exact solution* of the problems we dealt with (although the complexities usually are the same). Compare, for example, our computation times with the times given in [8]. Indeed, RelView is able to compute all maximum cliques within a reasonable time in the case of sparse graphs. As mentioned before in the consideration of random instances, however, it has its difficulties if density increases. But it should be emphasized that the system yields *all solutions*. In some applications this may be helpful. For example, the enumeration of maximum independent sets can be reduced to the enumeration of maximum cliques. Using the *edge adjacency relation* construction (see [12]), thus, we are able to enumerate all maximum respectively all perfect matchings of graphs. The latter can be used to compute permanents of 0/1-matrices, since this number equals the number of perfect matchings of a specific bipartite graph.

6 A Lattice-Theoretic Application: Cut Completion

Assume (X, R) to be a partially ordered set, that is, $R : X \leftrightarrow X$ to be a partial order on X. For a set $Y \in 2^X$ let $Y^{\downarrow}$ denote the set of its lower bounds and

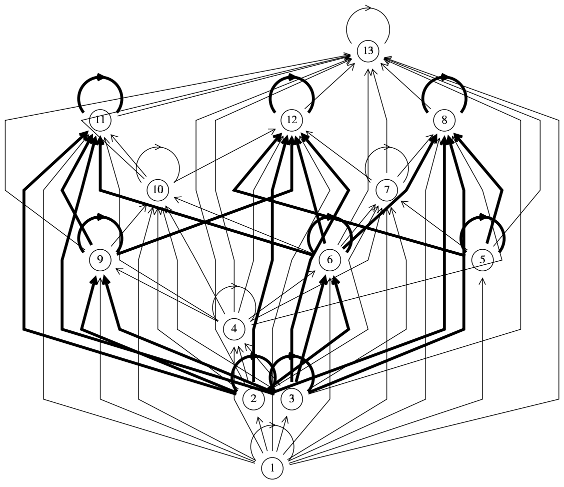


Fig. 4. Visualization of a cut completion

$Y^{\uparrow}$ denote the set of its upper bounds with respect to R. If $Y = Y^{\uparrow\downarrow}$, then this set is called a *Dedekind cut* of (X, R). Obviously, for each element $x \in X$ the set $[x] := \{y \in X \mid R_{yx}\}$ is a Dedekind cut of (X, R), called the *principal cut* generated by x. Now, let $\mathfrak{C}$ denote the set of all Dedekind cuts of (X, R) and $\mathfrak{P}$ be the subset of all principal cuts. Then $(\mathfrak{C}, \subseteq)$ is a complete lattice. It is called the *cut completion* (or Dedekind-MacNeille completion) of (X, R), since it contains $(\mathfrak{P}, \subseteq)$ as sub-order and the latter is order-isomorphic to (X, R) via the injective function $\sigma : X \to \mathfrak{C}$, mapping x to the principal cut generated by x. The lattice $(\mathfrak{C}, \subseteq)$ is the smallest complete lattice which embeds (X, R) as a sub-order; for more details on cut completion see [6].

Again we start with the vector-representation of the decisive set $\mathfrak{C}$. Assume $Y \in 2^X$. If we formalize lower and upper bounds using predicate logic and apply definition (3) of the membership relation $\mathsf{M} : X \leftrightarrow 2^X$, then we have

$$\begin{aligned} Y = Y^{\uparrow\downarrow} &\iff \forall\, x : x \in Y \leftrightarrow x \in Y^{\uparrow\downarrow} \\ &\iff \forall\, x : x \in Y \leftrightarrow (\forall\, y : y \in Y^{\uparrow} \to R_{xy}) \\ &\iff \forall\, x : x \in Y \leftrightarrow (\forall\, y : (\forall\, z : z \in Y \to R_{zy}) \to R_{xy}) \\ &\iff \forall\, x : \mathsf{M}_{x,Y} \leftrightarrow (\forall\, y : (\forall\, z : \mathsf{M}_{zY} \to R_{zy}) \to R_{xy})\,. \end{aligned}$$

Now, we apply the same strategy as in the case of cliques in combination with property (1) to replace in the last formula all logical constructions by relation-algebraic ones. Doing so, we arrive after some steps at

$$\mathit{cuts}(R) := (\mathrm{syq}(\mathsf{M}, \overline{\overline{R}\,\overline{\overline{R}}^{\mathsf{T}}\mathsf{M}}) \cap \mathsf{I})\mathsf{L} : 2^X \leftrightarrow \mathbf{1} \tag{8}$$

as relation-algebraic specification of the vector representing the set $\mathfrak{C}$ of all Dedekind cuts of (X, R), where $\mathsf{I} : 2^X \leftrightarrow 2^X$ and $\mathsf{L} : 2^X \leftrightarrow \mathbf{1}$. Using (8), the column-wise representation of the set $\mathfrak{C}$ by

$$\mathit{cutslist}(R) := \mathsf{M}\,\mathrm{inj}(\mathit{cuts}(R))^{\mathsf{T}} : X \leftrightarrow \mathfrak{C} \tag{9}$$

is an immediate consequence of the remark of Section 2. The same holds for the inclusion on the set of cuts, i.e., the partial order of the cut lattice $(\mathfrak{C}, \subseteq)$. Here we have the relation-algebraic specification

$$\mathit{cutord}(R) := \overline{\mathit{cutslist}(R)^{\mathsf{T}}\,\overline{\mathit{cutslist}(R)}} : \mathfrak{C} \leftrightarrow \mathfrak{C}\,. \tag{10}$$

Also the embedding of (X, R) into its cut completion $(\mathfrak{C}, \subseteq)$ can be formulated quite easily using relational algebra. Given $x \in X$ and $Y \in \mathfrak{C}$, we obtain

$$[x] = Y \iff \forall\, y : R_{yx} \leftrightarrow y \in Y \iff \mathrm{syq}(R, \mathit{cutlist}(R))_{xY}\,,$$

where the first step uses the definition of principal cuts and the second step applies (1) and the equivalence of $y \in Y$ and $\mathit{cutlist}(R)_{yY}$. This leads to

$$\mathit{sigma}(R) := \mathrm{syq}(R, \mathit{cutset}(R)) : X \leftrightarrow \mathfrak{C} \tag{11}$$

as relation-algebraic version of the embedding function $\sigma : X \to \mathfrak{C}$.

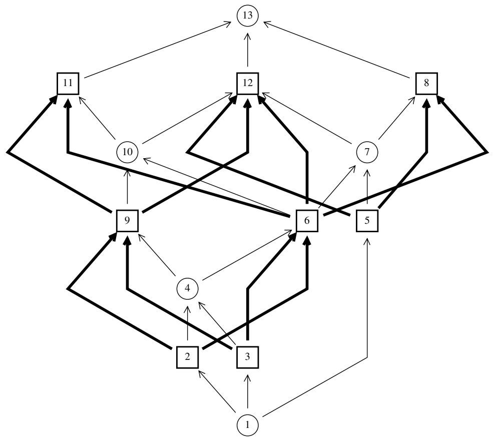


Fig. 5. Visulization of the *Hasse-diagram*

It is trivial to translate (8) until (11) into RelView-code. This allows to compute and visualize cut completions with the tool. Figure 4 graphically depicts the relation $C := \mathit{cutord}(R) : \mathfrak{C} \leftrightarrow \mathfrak{C}$ of the cut completion of an ordered set (X, R). The embedding into $(\mathfrak{C}, C)$ is visualized by boldface arrows, where the latter has been obtained by two steps. First, the sub-relation $S := \mathit{sigma}(R)^{\mathsf{T}} R\, \mathit{sigma}(R)$ of C has been computed. Then, in the graph-representation of C the arrows corresponding to elements of S have been marked using a specific command.

Layouts of partial orders as given in Figure 4 are neither economic nor easy to comprehend since they contain many superfluous arrows. Therefore, it is customary to depict only the Hasse-diagram as shown in Figure 5. To obtain this picture, we have computed the Hasse-diagrams H_C and H_S of C and S, respectively, via calls of a small RelView-program. Then we have marked the arrows corresponding to elements of H_S in the graph-representation of the union $H_C \cup H_S$. And, finally, we have emphasized the set of vertices represented by the vector $\mathit{sigma}(R)^{\mathsf{T}} \mathsf{L} : \mathfrak{C} \leftrightarrow \mathbf{1}$ by drawing these vertices as squares.

7 Conclusion

We have presented the OBDD-based specific purpose Computer Algebra system RelView for relational algebra, and have exhibited its use by two examples. The novelty and real attraction of our approach is the combination of OBDDs, relational algebra, visualization, and animation in an efficient and flexible software system. RelView uses only standard procedures for OBDD manipulation, which are available in any OBDD-package. Hence, any improvement in the OBDD area leads to a greater efficiency of RelView computations.

Experience has taught us that relational algebra is a powerful tool for dealing with many problems on discrete structures. As the examples of Section 5 and 6 show, it is often possible to "calculate" concise algorithms from formal specifications, so that correctness is established by construction. Algorithms can be executed and their results visualized with RelView. This allows to check them against the specifications and to detect errors. Due to the shortness and clearness of relation-algebraic expressions respectively RelView programs, the user can easily play and experiment with them.

RelView has been combined with other tools, e.g., SniffAlyzer for viewing and analyzing software architectures facts (see [11]). We have isolated the core functionality of RelView from the entire system and collected in a C-library, called Kure. With this library at hand, relational algebra can efficiently and easily be integrated into many other software systems. The PetRA tool for the analysis of Petri nets is a first application of this approach; see [9] for details.

References

1. R. Behnke et al., RelView – A system for calculation with relations and relational programming, in: Proc. 1. Conf. on Fundamental Approaches to Software Engineering, LNCS 1382, Springer (1998), 318-321.
2. R. Berghammer, T. Hoffmann, Modelling sequences within the RelView system, J. of Univ. Comp. Sci. 7 (2001), 107-123.
3. R. Berghammer, B. Möller, G. Struth (eds.), Proc. 7th Int. Workshop *Relational Methods in Computer Science*, LNCS 3051, Springer (2004).
4. C. Brink, W. Kahl, G. Schmidt (eds.), Relational methods in computer science, Advances in Computer Science, Springer (1997).
5. R.E. Bryant, Symbolic Boolean manipulation with ordered binary decision diagrams, ACM Com. Surv. 24 (1992), 293-318.
6. B.A. Davey, H.A. Priestley, Introduction to lattices and orders, Cambridge Univ. Press (1991).
7. DIMACS implementation challenges (second challenge, 1992-1993), Available via URL http://dimacs.rutgers.edu/Challenges/.
8. T. Fable, Simple and fast: Improving a branch-and-bound algorithm for maximum clique, in: Proc. 10. Europ. Symp. on Alg., LNCS 2461, Springer (2002), 485-498.
9. A. Fronk, Using relational algebra for the analysis of Petri nets in a CASE tool based approach, Proc. 2nd IEEE Int. Conf. on Software Engineering and Formal Methods, IEEE Press (2004), 396-405.
10. B. Leoniuk, ROBDD-based implementation of relational algebra with applications (in German), Diss., Univ. Kiel (2001).
11. U. Milanese, On the implementation of a ROBDD-based tool for the manipulation and visualization of relations (in German), Diss., Univ. Kiel (2003).
12. G. Schmidt, T. Ströhlein, Relations and graphs. Discrete mathematics for computer scientists, EATCS Monographs on Theoret. Comp. Sci., Springer (1993).
13. I. Wegener, Branching programs and binary decision diagrams: Theory and applications, SIAM Monographs on Discr. Math. and Appl., SIAM (2000).

On the Use of Gröbner Bases for Computing the Structure of Finite Abelian Groups

M. Borges-Quintana[1,⋆], M.A. Borges-Trenard[1], and E. Martínez-Moro[2,⋆⋆]

[1] Dpto. de Matemática, FCMC, U. de Oriente, Santiago de Cuba, Cuba
mijail@mbq.uo.edu.cu, mborges@mabt.uo.edu.cu
[2] Dpto. de Matemática Aplicada, U. de Valladolid, Spain
edgar@maf.uva.es

Abstract. Some algorithmic properties are obtained related with the computation of the elementary divisors and a set of canonical generators of a finite abelian group, this properties are based on Gröbner bases techniques used as a theoretical framework. As an application a new algorithm for computing the structure of the abelian group is presented.

1 Introduction

Knowing structure of finite abelian groups has many applications on discrete applied mathematics (see [15] and the references within for some applications). Compute the structure of a finite abelian group can be reduced to computing the Smith Normal Form (SNF) of integer matrices (see, for instance, [5]). There are many works devoted to the computation of the SNF, the Hermite Normal Form (HNF) and related problems (the standard algorithm is Gaussian elimination over the integers), some approaches (not an exhaustive list) can be found in [9,7,10]. Some interesting current implementations of algorithms for computing the SNF can be found in [11]. The best known complexities seem to appear in [13,14] for a deterministic algorithm or in [8] for a probabilistic algorithm of Monte Carlo type. Finally, in [6] the authors give a number of algorithms for computing with Abelian groups, such as computing kernels, inverse images, images, quotients, extensions, etc; in each case, the authors choose the most suitable representation for the input and output, but the SNF is always around. In the papers [4,16,17] there is another direction for computing the group structure, which considers the group given in such a fashion that multiplication of elements, computing inverses, and comparing elements are possible. Our contribution is to clarify the relation of those problems with the shape and computation of a certain Gröbner basis and to use that relation in order to efficiently compute finite abelian group structures.

⋆ This author is partially supported by Spanish MEC grant SB 2003-0287.
⋆⋆ Partially supported by Spanish MEC MTM2004-00876 and MTM2004-00958.

2 Preliminaries

Let $\mathcal{A}$ be a finitely generated Abelian group (group for short in the sequel). Let (Γ, M), be a presentation of $\mathcal{A}$, i.e. a pair given by $\Gamma := \{\alpha_1, \dots, \alpha_n\}$, a generating set of $\mathcal{A}$, and $M := \{m_{ij}\}$, a $k \times n$ matrix, so that the following sequence is exact:

$$\mathbb{Z}^k \xrightarrow{M} \mathbb{Z}^n \xrightarrow{\Gamma} \mathcal{A} \longrightarrow 1.$$

A presentation of $\mathcal{A}$ can also be described in terms of a quotient of the corresponding free abelian monoid, namely, let Γ^{-1} be the set of formal inverses, then $\mathcal{A}$ is represented by:

$$\langle \Gamma \cup \Gamma^{-1} \mid \sigma \rangle = \left\langle \Gamma \cup \Gamma^{-1} \mid \prod_{j=1}^{n} \beta_j^{|m_{ij}|} = 1, i \in [1,k];\ \alpha_j \alpha_j^{-1} = 1, j \in [1,n] \right\rangle,$$

where $\beta_j = \alpha_j$ if $m_{ij} \geq 0$ or $\beta_j = \alpha_j^{-1}$ if $m_{ij} < 0$. From σ it is usual derived the following set of binomials:

$$\mathrm{P}(\sigma) := \left\{ \prod_{j=1}^{n} \beta_j^{|m_{ij}|} - 1, i \in [1,k];\ \alpha_j \alpha_j^{-1} - 1, j \in [1,n] \right\}.$$

On the other hand, given a subset $\mathcal{S}$ of $\mathcal{A}$, we will denote by $\langle \mathcal{S} \rangle$ (as usual) to the subgroup generated by $\mathcal{S}$. We will make use with the commutative polynomial ring $K[\alpha_1, \dots, \alpha_n, \alpha_1^{-1}, \dots, \alpha_n^{-1}]$, where K is a field. In this context, $\mathit{Ideal}(F)$ stands for the ideal generated by the subset of polynomials F. The problem addressed in this paper is the following:

Problem 1 (Computing the structure of finite abelian groups.).
Given (Γ, M), *a presentation of a finite Abelian group* $\mathcal{A} \neq \{1\}$,
Compute *positive integers* $m_1, \dots, m_k$, *with* $m_1 > 1$, $m_i \mid m_{i+1}$, $1 \leq i < k$ *and an isomorphism* $\phi : \mathcal{A} \longrightarrow \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$, *which is given in terms of the images of the generators.*

Note that this is a much harder computational problem than just to determine the invariants (elementary divisors) of $\mathcal{A}$, $m_1, \dots, m_k$ .In the general case, it may be possible that either M does not represent a finite group or M stands for the trivial group, but it can be possible to discard both situations as a part of the algorithm devoted to solve the problem of group structure computation.

3 Gröbner Bases Associated to Finite Abelian Groups

The reader can find a basic background on Gröbner Bases in [3,12]. We will make use the lexicographical term ordering induced by $\alpha_1 \prec \cdots \prec \alpha_n \prec \alpha_1^{-1} \prec \cdots \prec \alpha_n^{-1}$ and defined as:

$$\alpha_1^{a_1} \cdots \alpha_n^{a_n} (\alpha_1^{-1})^{a_{n+1}} \cdots (\alpha_n^{-1})^{a_{2n}} \prec \alpha_1^{b_1} \cdots \alpha_n^{b_n} (\alpha_1^{-1})^{b_{n+1}} \cdots (\alpha_n^{-1})^{b_{2n}} \iff$$
$$\iff \exists\, j \mid a_j < b_j \text{ and } a_i = b_i, \text{ for } i > j.$$

Definition 1 (Least Exponent). *Let $\mathcal{A} \neq \{1\}$ be a group; $\xi_1, \ldots, \xi_r, \delta$ elements of $\mathcal{A}$. The least exponent of δ with respect to $\{\xi_1, \ldots, \xi_r\}$ is the least natural number m such that $\delta^m \in \langle \{\xi_1, \ldots, \xi_r\} \rangle$.*

We will represent m by $\mathcal{LE}(\delta, \{\xi_1, \ldots, \xi_r\})$. It is clear that

$$\mathcal{LE}(\delta, \{\xi_1, \ldots, \xi_r, \xi_{r+1}\}) \mid \mathcal{LE}(\delta, \{\xi_1, \ldots, \xi_r\}). \tag{1}$$

Theorem 1 (Characterization of Finite Abelian Groups). *Let $\mathcal{A}$ be a group, (Γ, M) a presentation of $\mathcal{A}$, $\mathrm{P}(\sigma)$ as in Section 2. Then $\mathcal{A}$ is finite if and only if the reduced Gröbner basis of $Ideal(\mathrm{P}(\sigma))$, with respect to $\prec$, has the following shape:*

$$rGb(P(\sigma)) := \left\{ \alpha_i^{m_i} - \prod_{j=1}^{i-1} \alpha_j^{n_{ij}} \mid i \in [1,n] \right\} \bigcup \left\{ \alpha_i^{-1} - \prod_{j=1}^{i-1} \alpha_j^{k_{ij}} \mid i \in [1,n] \right\}, \tag{2}$$

where n_{ij}, k_{ij} are not negative integers less than m_j, m_1 is the order of α_1 and, for all $i \in [2,n], m_i = \mathcal{LE}(\alpha_i, \{\alpha_1, \ldots, \alpha_{i-1}\})$. Moreover, $\mathcal{A}$ is the trivial group if and only if $rGb(P(\sigma)) = \bigcup_{i=1}^{n} \{\alpha_i - 1, \alpha_i^{-1} - 1\}$.

Proof (Sketch, details are left to the reader). The binomials $\alpha_i^{m_i} - \prod_{j=1}^{i-1} \alpha_j^{n_{ij}}$ and $\alpha_i^{-1} - \prod_{j=1}^{i-1} \alpha_j^{k_{ij}}$ belong to $Ideal(P(\sigma))$; $\alpha_i^{m_i}$ and α_i^{-1} are respectively the maximal terms with respect to $\prec$. Furthermore, if we assume that a binomial different from the above ones belong to $rGb(P(\sigma))$ and that the group is finite, then either we will get a contradiction with the definition of m_i or the binomial can be reduced by means of the above set of binomials.

Remark 1.

1. Compute $rGb(P(\sigma))$ is in essence to compute the HNF of the initial matrix M (See Section 4). In fact, it can be possible to compute the structure of $\mathcal{A}$ by means of typical Gröbner bases techniques, namely: Compute the Gröbner basis of $Ideal(P(\sigma))$, for deciding whether $\mathcal{A}$ is a not trivial finite group; if so, introduce systematically new "convenient" generators and corresponding relations, then update the Gröbner basis (using the elimination property), and so forth, until the "new" Gröbner basis "reveals" the structure. We do give more details because this approach may lead to a more complex way than the already known methods. Our intention is rather to use the shape of the above Gröbner basis in order to devise an efficient way for computing group structures.
2. In practical terms, we will not work from now on with the relations related to the inverses ($\alpha_i^{-1} - \prod_{j=1}^{i-1} \alpha_j^{k_{ij}}$, $i \in [1,n]$).

Proposition 1 (Computing the least exponent). *Let $w := \prod_{i=1}^{s} \alpha_i^{k_i}$, with $k_s \neq 0$, then $\mathcal{LE}(w, \{\alpha_1, \ldots, \alpha_{s-1}\}) = \dfrac{m_s}{GCD(m_s, k_s)}$.*

Proof. Let $t := \frac{m_s}{GCD(m_s,k_s)}$ and $m := \mathcal{LE}(w, \{\alpha_1, \ldots, \alpha_{s-1}\})$. Let us take $m = qt+r$, where q, r are integers and $r \in [0, t-1]$. Then, $w^m = \prod_{i=1}^{s-1}(\alpha_i^{k_i m})\alpha_s^{k_s qt}\alpha_s^{k_s r}$, which implies that $\alpha_s^{k_s r} \in \langle\{\alpha_1, \ldots, \alpha_{s-1}\}\rangle$; hence $r = 0$ (t is the least natural number such that $m_s \mid k_s t$). Now, as $m = qt$ and we are interested in the least number, we infer that $m = t$.

4 Hermite Normal Form and Gröbner Bases

There is not a unique definition of HNF. The following definition is given in [10].

Definition 2 (Hermite normal form). *A nonsingular square integer matrix is in HNF if it is upper triangular with positive diagonal elements. Further, each off-diagonal element is nonpositive and strictly less in absolute value than the diagonal element in its column.*

If the group is finite, the HNF could be assumed as square matrix. If it results a $k \times n$ matrix ($k > n$), only the first n rows would be nonzeros. Given $(m_{i\,j})$, a HNF representation of a finite group, assume that the columns correspond to the generators of the group in decreasing ordering (i.e. the first columns corresponds to α_n). Translating the HNF to the corresponding set of binomials $\{\alpha_i^{m_{i\,i}} - \prod_{j=1}^{i-1}\alpha_j^{-m_{i\,j}}\}$, one can see that it has the form given in (2) for the reduced Gröbner basis, removing the set of binomials corresponding to the inverse of the generators (see Theorem 1). On the other hand, starting from the set of binomials of $rGb(P(\sigma))$ related with the generators $\alpha_1, \ldots, \alpha_n$, one can get the associeted echelon form matrix that represents the group. The characteristic of the exponents in (2) ensures that this matrix is in HNF.

In intermediate calculus of many algorithms a bound of the order of the group is used in order to decrease the size of the coefficients. It is well known that the order of the group is bounded by any determinant of an square submatrix of a matrix that represent the group. Let us denote by d the determinant of the main square submatrix of M (after an arranment of the order of the generators if it were necessary). A bound for d is $\mathsf{d_b} = (\sqrt{n}\,\|M\|)^n$, where $\|M\|$ denotes the maximum magnitud of entries in the initial matrix M.

From now on we will denote by σ_1 the presentation of a finite group obtained from the translation of the computation of the HNF. It is also enough for our purpose to compute the upper triangular matrix, where the n_{ij}'s do not have to be reduced with respect to the corresponding m_j but just with respect to d. After the computation of σ_1, it is easy to see that the order of $\mathcal{A}$ can be computed as $o(\mathcal{A}) := \prod_{i=1}^{n} m_i$.

Example 1 (A HNF for a finite group and a presentation σ_1). Given the finite group $\mathcal{A}$ generated by a, b, c, d and presented by the matrix

$$M := \begin{vmatrix} 10 & -3 & -3 & 0 \\ 30 & 6 & -3 & -9 \\ 20 & -6 & 2 & -6 \\ 70 & -21 & 11 & -6 \end{vmatrix}, \text{ where the order of the generators is } a \prec b \prec c \prec d.$$

The HNF of M is given by

$H := \begin{vmatrix} 10 & -3 & -3 & 0 \\ 0 & 15 & -2 & -3 \\ 0 & 0 & 8 & -6 \\ 0 & 0 & 0 & 18 \end{vmatrix}$, from where σ_1 can be obtained as $\sigma_1 := \{a^{18} = 1,\, b^8 = a^6,\, c^{15} = a^3 b^2,\, d^{10} = b^3 c^3\}$.

The order of the group can be computed, $o(\mathcal{A}) = 18 \cdot 8 \cdot 15 \cdot 10 = 21600$. In this case, d coincide with the order of the group. However $\mathsf{d_b} = \left(\sqrt{4}\,70\right)^4 = 38416 \cdot 10^4$.

5 Elementary Divisors and Generators

Assume that we have already computed a presentation σ_1 of $\mathcal{A}$, generated by $\alpha_1, \ldots, \alpha_n$.

$$\sigma_1 = \left\{ \alpha_i^{m_i} = \prod_{j=1}^{i-1} \alpha_j^{n_{ij}} \mid i \in [1, n] \right\}, \tag{3}$$

where n_{ij}, k_{ij} are not negative integers less than m_j, m_1 is the order of α_1 and, for all $i \in [2, n], m_i = \mathcal{LE}(\alpha_i, \{\alpha_1, \ldots, \alpha_{i-1}\})$. Let us assume we have computed the firts s elementary divisors and a corresponding set of canonical generators $\beta_1, \ldots, \beta_s$. Then σ_{s+1} will be the updated presentation of $\mathcal{A}$ with the same characteristic of (3) but for the order $\beta_1 \prec \ldots \prec \beta_s \prec \alpha_1 \prec \ldots \prec \alpha_n$.

In this section we show how to compute, given the presentation σ_s, the next elementary divisor and a canonical generator β_s associated. The shape of the presentations σ_s's will play an essential role in the proofs and procedures of this section. By $RF(\alpha, \sigma)$ we denote the reduced form of the element α module σ . If the presentation σ is clear from the context we will use just $RF(\alpha)$.

Definition 3 (Less canonical exponent). *Given $\alpha \in \mathcal{A}$, we define $\ell_s(\alpha) := \mathcal{LE}(\alpha, \{\beta_1, \ldots, \beta_s\})$. The* **greatest canonical exponent is** $m(s) := \max\{\ell_s(\xi) \mid \xi \in \mathcal{A}\}$.

In the Theorem below it is shown that $m(s)$ is the s-th elementary divisor. In order to compute the less canonical exponent of an element, the presentation σ_s is needed. In the sequel these two procedures will be shown in details.

Theorem 2 (Computing the next canonical generator). *Supose there are given the first s canonical generators and the presentation σ_{s+1} of $\mathcal{A}$. Let m be the least common multiple (LCM) of $\ell_s(\alpha_1), \ldots, \ell_s(\alpha_n)$. Then, if $m = 1$, the s canonical generators already introduced generate the group $\mathcal{A}$; otherwise, if $m \neq 1$, the following three propositions are satisfied:*

i. There exists $\beta_s^* \in \mathcal{A}$ ($\beta_s^* = \prod_{i=1}^n \alpha_i^{q_i}$) such that $m = \ell_s(\beta_s^*)$.
ii. For $s = 0$, $\beta_1 = \beta_0^*$ is such that $o(\beta_1) = m = m(0)$.
iii. For $s \geq 1$, let ${\beta_s^*}^m = \prod_{j=1}^s \beta_j^{a_j}$, with $0 \leq a_j < o(\beta_j)$, $j \in [1, s]$. Then, for all $j \in [1, s]$, $m \mid GCD(o(\beta_j), a_j)$.
Moreover, the element $\beta_{s+1} = \prod_{j=1}^s \beta_j^{t_j} \prod_{i=1}^n \alpha_i^{q_i}$, $t_j = \frac{(o(\beta_j) - a_j)}{m}$, $j \in [1, s]$, satisfies the following conditions:

1. $o(\beta_{s+1}) = m = m(s)$,
2. $\langle\beta_1, \ldots, \beta_s\rangle \cap \langle\beta_{s+1}\rangle = \{1\}$.

Proof. If $m = 1$,for all $i \in [1, n]$, we have that $\ell_s(\alpha_i) = 1$, i.e., the initial generators α_i's are already superfluous, which means that the group $\mathcal{A}$ is generated by the set of canonical generators $\{\beta_1, \ldots, \beta_s\}$.

Let $m > 1$. Let us prove first that for $s = 0$ the propositions of the theorem are satisfied. Obviously, $\ell_0(\alpha_i)$ corresponds to the order of each generator. It is well known from the Group Theory (not only for abelian groups) that it can be obtained an element β_0^* of order $m = LCM(\ell_0(\alpha_i) \mid i = [1, n])$ (in Section 6 Algorithm 1 shows how to compute such an element). Note that this element β_0^* satisfies the Proposition *i*, since with respect to the trivial subgroup, the least exponent of any element coincides with its order ($m = o(\beta_0^*) = \mathcal{LE}(\beta_0^*, \{1\})$).

It is clear that $w^m = 1$ (for all $w \in \mathcal{A}$), therefore, $o(w) \mid m$ (for any $w \in \mathcal{A}$), which implies that $m(0) \leq m$. Analyzing Definition 3 for $s = 0$, it is obvious from the construction of β_0^* that $o(\beta_0^*) = m = m(0)$. Thus $\beta_1 = \beta_0^*$; and Proposition *ii* is proved.

Now let us assume that the theorem holds for the first s canonical generators ($s \geq 1$); particularly, we have:

$$o(\beta_{j+1}) = \mathcal{LE}(\beta_{j+1}, \{\beta_1, \ldots, \beta_j\}) = LCM(\ell_j(\alpha_i) \mid i \in [1, n]) = m(j), \quad \text{for all } j \in [0, s-1], \tag{4}$$

let us prove that the conditions are satisfied also for $s + 1$. We compute first $\ell_s(\alpha_i)$ (see Algorithm 2 in Section 6) for any non superflous α_i such that we get:

$$\alpha_i^{\ell_s(\alpha_i)} = \prod_{j=1}^{s} \beta_j^{k_{ij}}. \tag{5}$$

Following the same idea of the case $s = 0$, it can be proved that $m = m(s)$. Using the Algorithm 1 (see Sección 6) we get β_s^* such that:

$$\beta_s^* = \prod_{i=1}^{n} \alpha_i^{q_i}, \qquad {\beta_s^*}^m = \prod_{j=1}^{s} \beta_j^{a_j}, \quad 0 \leq a_j < o(\beta_j), \;\; j \in [1, s], \tag{6}$$

where $m = \ell_s(\beta_s^*)$. It follows that condition *i* of the theorem holds; as well as the equality $m = m(s)$ of *iii*.1. On the other hand, the condition $m \mid a_j$ in *iii* is a consequence of Lema 1 (see Section 6), whereas $m \mid o(b_j)$ follows from (1) and (4).

Consider

$$\beta_{s+1} = \beta_s^* \prod_{j=1}^{s} \beta_j^{(o(b_j)-a_j)/m} \tag{7}$$

It is easy to prove that $\beta_{s+1}^m = 1$, then $o(\beta_{s+1}) \leq m$; besides, ${\beta_s^*}^k \notin \langle\beta_1, \ldots, \beta_s\rangle$ if $k < m$, consequently, $o(\beta_{s+1}) = m$.

Therefore, the chain of equalities of Proposition *iii*.1 are satisfied $o(\beta_{s+1}) = m = m(s)$. In addition, note that from the equality $o(\beta_{s+1}) = m$, we have that $\mathcal{LE}(\beta_{s+1}, \{\beta_1, \ldots, \beta_s\}) = o(\beta_{s+1})$ and Proposition *iii*.2 follows.

6 Others Results and Algorithms

In this section we show the results and algorithms which allow to compute the canonical generators following the results in Theorem 2. First, Lemma 1 proves that the t_j's of 2.*iii* are well defined, as well as, Algorithm 1 compute β_s^*. In Subsection 6.2, some comments are given to show how to compute σ_{s+1}, finally, Algorithm 2 is presented in Subsection 6.3 in order to compute the lest canonical exponents $\ell_s(\alpha_i)$ for each initial generator α_i of the group $\mathcal{A}$.

Lemma 1. *Let c be an element of $\mathcal{A}$ and $\beta_1, \ldots, \beta_s$ $(s \geq 1)$ a set of elements of $\mathcal{A}$ that satisfy (4). Let $k = \mathcal{LE}(c, \{\beta_1, \ldots, \beta_s\})$, If we have the equality: $c^k = \prod_{j=1}^{s} \beta_j^{a_j}$, then $k \mid a_j$ for all $j \in [1, s]$.*

Proof. Let us assume that for i elements $(i < s)$ the lemma holds, and let us prove that it holds for $i = s$.

Suppose, without lost of generality, that $a_s \neq 0$. Let p be any prime factor of k. By $\mathcal{GP}(m, p)$ we denote the greatest power of p which is a factor of m, in the case p is not a divisor of m, $\mathcal{GP}(m, p) := 1$ (p^0), on the other hand, for zero we have $\mathcal{GP}(0, p) := p^{+\infty}$.

Let $\mathcal{GP}(k,p) = p^{tc}$, $\mathcal{GP}(a_j,p) = p^{b_j}$ $\mathcal{GP}(o(\beta_j),p) = p^{t_j}$, con $j \in [1, s]$. As a direct consequence of Proposition 1, the number $k_s := \frac{o(\beta_s)}{GCD(a_s,\, o(\beta_s))}$ is the least exponent for c^k and $\langle \beta_1, \ldots, \beta_{s-1} \rangle$.

Lifting to k_s in both sides of the equality of the lemma we get

$$c^{m_s} = \prod_{j=1}^{s-1} \beta_j^{a_{s_j}}, \text{ where } m_s = k\, k_s = \mathcal{LE}(c, \{\beta_1, \ldots, \beta_{s-1}\})$$

and $a_{s_j} = a_j\, k_s,\ \ j \in [1, s-1]$.

If the set $\{\beta_1, \ldots, \beta_s\}$ satisfies (4), the set $\{\beta_1, \ldots, \beta_{s-1}\}$ satisfies also (4); then, we can apply the induction over $i = s - 1$ obtaining:

$$m_s \mid a_{s_j}, \text{ for all } j \in [1, s-1]. \tag{8}$$

Taking into account (8), $\mathcal{GP}(m_s, p) = \mathcal{GP}(k\, k_s, p) \leq \mathcal{GP}(a_j k_s, p)$ for all $j \in [1, s-1]$, from where it is obtained that $\mathcal{GP}(k, p) \leq \mathcal{GP}(a_j, p)$; consequently,

$$t_c \leq b_j, \text{ for all } j = [1, s-1]. \tag{9}$$

We have also that $m_s \mid m(s) = o(\beta_s)$ (see (4)), then

$$\mathcal{GP}(m_s, p) \leq \mathcal{GP}(o(\beta_s), p). \tag{10}$$

With respect to a_s let us divide the analysis in two cases:

$$(1)\quad \mathcal{GP}(a_s, p) \geq \mathcal{GP}(o(\beta_s), p), \qquad (2)\quad \mathcal{GP}(a_s, p) < \mathcal{GP}(o(\beta_s), p).$$

For case 1, we have $\mathcal{GP}(m_s, p) = p^{tc}$, and because of (10) it follows that

$$t_c \leq b_s. \tag{11}$$

For case 2, $\mathcal{GP}(m_s, p) = p^{tc + ts - bs}$. Due to (10) the following inequality holds

$$p^{tc + ts - bs} \leq p^{ts}; \tag{12}$$

therefore, $t_c \leq b_s$. Analyzing the previous inequality and (9), (11), it has been obtained that:

$$t_c \leq b_j \quad j \in [1, s].$$

It means, for any prime factor p of k

$$\mathcal{GP}(k, p) \leq \mathcal{GP}(a_j, p), \text{ for all } j \in [1, s],$$

As a consequence, $k \mid a_j$, for all $j \in [1, s]$.

Algorithm 1. (Computing β_s^*)
Input: *σ_s and $\ell_s(\alpha_i)$ $i = 1, \dots, n$.*
Output: *The element β_s^* and $m(s) = LCM(\{\ell_s(\alpha_i) \mid\in [1, n]\})$.*

```
1. $k := max\{i \in [1, n] \mid m_i \neq 1\}$;
2. $\beta_s^* := \alpha_k$, $m := \ell_s(\alpha_k)$; k:=k-1
3. While ( $k \geq 1$ ) do
4.      If ($m_k \neq 1$) then
5.        $p := GCD(m, \ell_s(\alpha_k))$
6.          If ($p \neq \ell_s(\alpha_k)$) and ($p \neq m$) then
7.            $p_1 := p$, $t := m/p_1$, $p_2 := GCD(p_1, t)$;
8.              While ($p_2 \neq 1$) do
9.                $p_1 := p_1/p_2$, $p_2 := GCD(p_1, t)$;
10.           $m := (m\,\ell_s(\alpha_k))/p$, $\beta_s^* := \beta_s^{*p_1}\,\alpha_k^{p/p_1}$;
11.         else  If ($p = m$) then $\beta_s^* := \alpha_k$, $m := \ell_s(\alpha_k)$;
12.     $k := k - 1$;
13. Return($RF(\beta_s^*), m$)
```

Correctness: The algorithm is based on the following property of the LCM:

$$LCM(m, \ell_s(\alpha_k)) = \frac{(m\ell_s(\alpha_k))}{GCD(m, \ell_s(\alpha_k))}$$

which gives a recursive formula for the computation of m (see Step 5 and 10); on the other hand, in Step 10 it is recomputed the element β_s^* such that its least canonical exponent corresponds to the updated m. This new element β_s^* is obtained as a product of a power of the former β_s^* ($\beta_s^{*p_1}$) and a power of α_k (${\alpha_k}^{p/p_1}$) such that their least canonical exponents are relative primes and their product is equal to m;[1] the computation of the corresponding powers of β_s^* and α_k is done in the **While** of Step 8, which performs Step 9 iteratively.

If the generator α_k is superfluous it is directly taken the next generator (see the condition in Step 4 and Step 12).

If $LCM(m, \ell_s(\alpha_k)) = \ell_s(\alpha_k)$, which means $GCD(m, \ell_s(\alpha_k)) = m$ (see the condition of Step 11), then $m = \ell_s(\alpha_k)$ and $\beta_s^* = \alpha_k$ (see Step 11). In the case that $LCM(m, \ell_s(\alpha_k)) = m$, the values of m and β_s^* remain unchange, and the next generator is taken.

[1] The product of two elements whose least canonical exponents are relative prime, has least canonical exponent equal to the product of those exponents.

6.1 A Complete Example

Let $\mathcal{A}$ be the finite group defined in the Example 1. First, let us compute an element β_1 of maximum order. The element β_0^* computed by Algorithm 1 is exactly the first canonical generator β_1 (see Theorem 2.*ii*), then we will use directly β_1 instead of β_0^*. Let us compute first $\ell_0(\alpha_i) = o(\alpha_i)$, for $\alpha_i \in \{ a, b, c, d\}$

$$\ell_0(a) = 18,\ \ell_0(b) = 24,\ \ell_0(c) = 60,\ \ell_0(d) = 400.$$

The algorithm starts with d (see Step 1), $\beta_1 := d$, $m := \ell_0(d) = 400$.

The next generator c is taken, $\ell_0(c) \neq 1$, so, the condition in Step 4 is satisfied and the Steps from 5 to 10 are computed:
$p := GCD(m, \ell_0(c)) = GCD(400, 60) = 20$, as $p \neq m$ and $p \neq \ell_0(c)$ (see Step 6) then, $p_1 := p = 20$, $t := m/p_1 = 20$, $p_2 := GCD(p_1, t) = 20$, we have $p_2 \neq 1$ (see Step 8) therefore, following Step 9: $p_1 := p_1/p_2 = 1$, $p_2 := GCD(p_1, t) = 1$. When $p_2 = 1$, the **While** at Step 8 has ended, now the values of m and β_1 are updated according to Step 10:
$m := (m\,\ell_0(c))/p = 400 \cdot 3 = 1200$, $\beta_1 := \beta_1^{p_1}\, c^{p/p_1} = \beta_1\, c^{20} = d\, c^{20}$.

Taking now b, $p := GCD(m, \ell_0(b)) = GCD(1200, 24) = 24$, in this cases $p = \ell_0(b)$, thus b is not necessary in order to construct β_1. [2]

For a we have $p := GCD(m, \ell_0(a)) = GCD(1200, 18) = 6$, then, p satisfies the condition of Step 6, thus we do $p_1 := p = 6$, $t := m/p = 200$, $p_2 := GCD(p_1, t) = 2$, $p_2 \neq 1$; therefore, the **While** of Step 8 is performed. $p_1 := p_1/p_2 = 6/2 = 3$, $p_2 := GCD(p_1, t) = 1$; in Step 10 the new values of β_1 and m are computed: $m := (m\,\ell_0(a))/p = 1200 \cdot 3 = 3600$, $\beta_1 := \beta_1^3\, a^{6/3} = (c^{20}\, d)^3\, a^2 = a^2 c^{60} d^3$,

$$RF(\beta_1) = RF(a^2 c^{60} d^3) = a^2\, d^3.$$

Now we compute σ_2 with the order of the generators $\beta_1 \prec a \prec b \prec c \prec d$. In the subsection 6.2 we will give some comments about how to compute the new presentation, meanwhile, let us assume we have computed σ_2. Note that in the context of the Buchberger Algorithm [3] it would be enough to compute the reduced Gröbner basis of the ideal generated by $P(\sigma_1) \cup \{\beta_1 - a^2 d^3\}$ with respect to the lexicographic ordering such that $\beta_1 \prec a \prec b \prec c \prec d$. The resulting set of binomials will correspond to $P(\sigma_2)$.

$$\sigma_2 = \{\beta_1^{3600} = 1,\ a^2 = \beta_1^{2800},\ c^3 = \beta_1^{220} a\}.$$

The generators b and d have been excluded from σ_2 because they become supefluous. Then Algorithm 1 will be apply to obtain β_1^*. Let us see first the expression of the equalities defined in (5), corresponding to the computation of $\ell_1(\alpha_i)$, for $\alpha_i \in \{a, c\}$. The computation of the $\ell_s(\alpha_i)$ is another subroutine needed for the computation of the elementary divisors (see Section 6.3).

$$a^2 = \beta_1^{2800}, \qquad c^6 = \beta_1^{3240} = \beta_1^{-360},$$

Both equalities show the result stated in Lemma 1 ($2 \mid 2800$ and $6 \mid 3240$).

[2] In Step 6 there are only two situations in which some computations have to be done: $p \neq m$ and $p \neq \ell_0(b)$, and $p = m$ (see Step 11). In the case $p = \ell_0(b)$ no computation is done and therefore the next generator is taken, keeping β_1 and m already computed until this moment.

Starting with the generator c, $m = 6$ and $\beta_1^* = c$. In the next step we have $p = \ell_1(a)$, then the values of m and β_1^* do not change (see the proof of the correctness of Algorithm 1). Therefore, $\beta_1^* = c$ and the relation corresponding to (6) is $c^6 = \beta_1^{3240}$.
Applying Theorem 2.*iii* we can compute the canonical generator β_2

$$\beta_2 = \beta_1^{(3600-3240)/6}\, c = \beta_1^{60}\, c.$$

Writting β_2 depending of the initial generators we obtain $\beta_2 = a^9b^4c^{10}$. Note that $o(\beta_1) \cdot o(\beta_2) = 3600 \cdot 6 = 21600 = o(\mathcal{A})$, which implies that in the next presentation σ_3 all the initial generators will be superfluous; consequentely,

$$\mathcal{A} = \langle a^2\, d^3\rangle \times \langle a^9b^4c^{10}\rangle,$$

thus the canonical structure of $\mathcal{A}$ is $(\mathbb{Z}/_{3600}\mathbb{Z}) \times (\mathbb{Z}/_{6}\mathbb{Z})$.

6.2 Computing the Presentation $\boldsymbol{\sigma_{s+1}}$

Suppose it is given the presentation σ_s with the generators $\beta_1 \prec \ldots \prec \beta_{s-1} \prec \alpha_1 \prec \ldots \prec \alpha_n$. Then, given a new element β_s we want to compute the new presentation σ_{s+1} for the ordering $\beta_1 \prec \ldots \prec \beta_s \prec \alpha_1 \prec \ldots \prec \alpha_n$. As we said in the previous section $\sigma_s' = \sigma_s \cup \{\beta_s = \prod_{i=1}^{s-1} \beta_i^{b_i} \prod_{j=1}^{n} \alpha_j^{k_j}\}$ is also a presentation of the group $\mathcal{A}$. In order to compute σ_{s+1} it is enough to have an algorithm for solving the following problem: " **Given** the presentation σ_s' **compute** the presentation σ_{s+1}".

Note that there is one change of the order among the generators of both presentations.The presentation σ' corresponds to an upper triangular matrix, if we move the last relation to position s and the rest of the ordering is kept invariant, then the new corresponding matrix it is not upper triangular, and for computing σ_{s+1} it would be enough to apply an algorithm in order to compute the corresponding HNF. Because there was just only one twist from an upper triangular matrix, the algorithm for recovering the HNF should be simpler in this case. In order to get benefit of the specific case for the starting matrix, the following proposition would play an important role and shows how to compute the new less exponent for the generators α_i's.

Proposition 2 (The new less exponent).
Let $\ell_i := \mathcal{LE}(\beta_s, \{\beta_1, \ldots, \beta_{s-1}, \alpha_1, \ldots, \alpha_i\})$ *and* $\beta_s^{\ell_i} = \prod_{j=1}^{s-1} \beta_j^{b_{ij}} \prod_{j=1}^{i} \alpha_j^{k_{ij}}$ *then*

$$\mathcal{LE}(\alpha_i, \{\beta_1, \ldots, \beta_{s-1}, \boldsymbol{\beta_s}, \alpha_1, \ldots, \alpha_{i-1}\}) = GCD(m_i, k_{ii}).$$

6.3 Computing $\boldsymbol{\ell_s(\alpha_i)}$

Algorithm 2. $\boldsymbol{(\ell_s(\alpha_i))}$
Input: $\sigma_{s+1} = \{\beta_i^{d_i} = 1 \mid i \in [1,s]\} \cup \{\alpha_i^{m_i} = \prod_{j=1}^{s} \beta_j^{h_j} \prod_{j=1}^{i-1} \alpha_j^{n_{ij}} \mid i \in [1,s]\}$.
Output: $\ell_s(\alpha_i)$ $i = 1, \ldots, n$.

```
1. $\ell_s(\alpha_1) = m_1,\ i := 2;$
2. While $(i \leq n)$ do
3.     If $m_i \neq 1$ then
4.         $l := i - 1,\ \ell_s(\alpha_i) := m_i,\ k_j := n_{ij}$ for $j = 1, \ldots, i-1;$
5.         While $(l \geq 1)$ do
6.             $t := GCD(k_l, m_l);$
7.             $\ell_s(\alpha_i) := \ell_s(\alpha_i) \cdot \frac{m_l}{t};$
8.             $k_j := \frac{k_j \cdot m_l}{t}$ for $1 \leq j \leq l;$
9.             $k_j := k_j + \frac{k_l}{m_l} \cdot n_{lj}$ for $1 \leq j \leq l-1;$
10.            $l := l - 1;$
11.    $i := i + 1;$
12. Return$(\{\ell_s(\alpha_i),\ i = 1, \ldots, n\})$.
```

7 Computing the Structure of the Abelian Group

Algorithm 3. (Computing the elementary divisors and a set of canonical generators)
Input: *A presentation σ_1 of the group $\mathcal{A}$ as it is given in (3).*
Output: *A set $\{m(i) \mid i \in [1,k]\}$, a set $\beta_1, \ldots, \beta_k$ of canonical generators of the group $\mathcal{A}$, a presentation σ_{k+1} of $\mathcal{A}$; such that, $\{m(i) = o(\beta_i) \mid i \in [1,k]\}$ are the elementary divisors of the group $(m(i+1) \mid m(i))$, $\mathcal{A} = \langle \beta_1 \rangle \times \ldots \times \langle \beta_k \rangle$, and σ_{k+1} is the presentation for $\mathcal{A}$ corresponding to the generators $\{\beta_1, \ldots, \beta_k, \alpha_1, \ldots, \alpha_n\}$ for the ordering $\beta_1 \prec \ldots \prec \beta_k \prec \alpha_1 \prec \ldots \prec \alpha_n$.*

```
1. $s := 0$.
2. Apply Algorithm 2 to compute $\ell_s(\alpha_1), \ldots, \ell_s(\alpha_n)$.
3. Apply Algorithm 1 to compute $m(s)$ and $\beta_s^*$ .
4. If $m(s) \neq 1$ then
5.     Compute $\beta_{s+1}$ by following Theorem 2.
6.     $s := s + 1$, compute the presentation $\sigma_{s+1}$.
7.     Goto Step 2.
8. Return$(\{m(s) \mid s \in [1,k]\}, \{\beta_s \mid s \in [1,k]\}, \sigma_{k+1})$.
```

Some Complexity Analysis: One advantage of this algorithm is that after the computation of σ_1 the order of the group is known; thus the bound for multiplication of integers of a bounded size can be improved, note that the common bound used is $\mathsf{d_b}$ which can be considerable higher than the order of the group (see Section 4). The loop of the algorithm, from Step 2 to Step 6, has as many iterations as the number of elementary divisors. Let be κ the number of such a divisors, it is obvious that n is a bound for κ but in many cases it results higher; for example, a cyclic group have just one elementary divisor. The basic operations of the algorithm are multiplication of integers (MULT), addition (SUM), greatest common divisor and extended greatest common divisor computation (GCD). Reduction of an integer module the order of the group have the same complexity of the operation MULT, so this operation is not taken into account because it does not increase the complexity. We use for this basic operations the complexity formulas given in [14], the complexity of computing σ_1 (equivalent to compute a HNF) is also taken from [14]. The complexity will be given in word operations which means that multiplication of integers can be done in constant time for the digits using a Residue Number System (RNS) representation of the integers[3], i.e. multiplying numbers of size bounded by d would take $\mathcal{O}(log\,\mathsf{d})$ word operations. Let $B(K) = M(K)log\,K$, where $M(K)$ denotes the best complexity to multiply integers of size 2^K, we will give the complexity in terms of this function $B(\cdot)$.

To perform Step 1 can be done in $\mathcal{O}(n^\theta log\,\mathsf{d_b} + n^2 log\,n B(log\,\mathsf{d_b}))$, where $\mathcal{O}(n^\theta)$ denotes the best complexity for a matrix multiplication algorithm (it is known that $2 < \theta \leq 3$, see [14] for an asymtotically better result).

Proposition 3 (Complexity). *Algorithm 3 computes the set of elementary divisors, a set of canonical generators and the corresponding canonical presentation in* $\mathcal{O}(\kappa\, n^3\, log\, o(\mathcal{A}) + \kappa\, n^2\, B(log\, o(\mathcal{A})) + \kappa\, n\, log\, o(\mathcal{A})\, B(log\, o(\mathcal{A})))$ *word operations.*

An advantage of this approach is the computation with integers of smaller size. The sizes of the integers in the computation of SNF are bounded by $\mathsf{d_b}$ and they usually grow up until that bound. However, after computation of σ_1, Algorithm 3 work with integers of size bounded by the order of the group, and $o(\mathcal{A}) \ll \mathsf{d_b}$. On the other hand, the calculus is more connected with the inside of the structure of the group since it depends on the number of elementary divisors. The complexity of the algorithm in terms of n, $\mathsf{d_b}$, and $B(\cdot)$ can be obtained by considering that $\kappa \leq n$ and $o(\mathcal{A}) \leq \mathsf{d_b}$, which will give a factor of n^5 while in [14] the best known complexity is given with a factor of $n^{\theta+1}$. Step 2 is the step of the algortihm of the highest complexity, $m(s)$ in Step 3 denotes the s-th elementary divisor.

Table 1. Complexity analysis

	basic operations	word operations
Step 2	$\mathcal{O}(n^2)\,GCD + \mathcal{O}(n^3)\,MULT$	$\mathcal{O}(n^2\,B(log\,o(\mathcal{A})) + n^3\,log\,o(\mathcal{A}))$
Step 3	$\mathcal{O}(n)(MULT\ +\ GCD)\,log\,m(s))$	$\mathcal{O}(n\ B(log\,o(\mathcal{A}))\,log\,m(s))$
Step 5	$\mathcal{O}(n^2)\,MULT$	$\mathcal{O}(n^2\,log\,o(\mathcal{A}))$
Step 6	$\mathcal{O}(n^2)\,MULT + \mathcal{O}(n)\,GCD$	$\mathcal{O}(n^2\,log\,o(\mathcal{A}) + n\,B(log\,o(\mathcal{A}))$

The main idea behind this algorithm is an iterative application of the elimination property of the Lexicographic ordering, whenever it is used that if an element β belongs to a certain subgroup $\langle\beta_1, \ldots, \beta_s\rangle$, where $\beta_i \prec \beta$, $i \in [1, s]$, then the reduced form of this element depends only on the elements $\{\beta_1, \ldots, \beta_s\}$. The algorithmic properties of Gröbner bases allow us to obtain Algorithm 3 by a strategic sequence of applications of this simple idea. Theorem 2, Lemma 1,

[3] See [14], where it is given also a conversion from word operations to obtain the true asymptotic bit complexity.

Algorithms 1 and 2 can be applied out of context of computing the canonical structure of the group; for example, in order to construct some subgroup when the group is given by generators and a multiplication.

References

1. M. Borges Quintana. *On some Gröbner Bases Techniques and their Applications.* (Spanish). Phd Thesis. Universidad de Oriente, Santiago de Cuba, Cuba. (2002)
2. B. Buchberger and F. Winkler. *Gröbner Bases and Applications.* Proc. of the International Conference "33 Years of Gröbner Bases". London Mathematical Society Series, V. 251, Cambridge University Press. (1998)
3. B. Buchberger. *Introduction to Gröbner Bases.* In [2], (1998) 3-31.
4. J. Buchmann, M.J. Jacobson, JR., and E. Teske. *On Some Computational Problems in Finite Abelian Groups.* Math. Comput., V. 66, N. 220, (1997) 1663-1687.
5. H. Cohen. *A Course in Computational Algebraic Number Theory.* (3rd corrected printing). V. 138 of Graduate Texts in Mathematics. New York, Springer. (1996)
6. H. Cohen, F. Díaz y Díaz, and M. Olivier. *Algorithmic Methods for Finitely Generated Abelian Groups.* J. Symbolic Computation 31, Issue 1-2, (2001) 133-147.
7. JG. Dumas, B.D. Saunders, and G. Villard.*On Efficient Sparse Integer Matrix Smith Normal Form Computations.* J. Symbolic Computation 32, Issue 1-2, (2001) 71-99.
8. W. Eberly, M. W. Giesbrecht, and G. Villard. *Computing the determinant and Smith form of an integer matrix.* In The 41*st* Annual IEEE Symposium on Foundations of Computer Science, Redondo Beach, CA. (2000)
9. G. Havas and B.S. Majewski. *Integer Matrix Diagonalization.* J. Symbolic Computation 24, Issue 3-4, (1997) 399-408.
10. C. S. Iliopoulos. *Worst-Case Complexity Bounds on Algorithms for Computing the Canonical Structure of Finite Abelian Groups and the Hermite and Smith Normal Forms of an Integer Matrix.* Siam J. Comput., vol. 18/4, (1989) 658-669.
11. F. Lübeck.*On the Computation of Elementary Divisors of Integer Matrices.* J. Symbolic Computation 33, Issue 1, (2002) 57-65.
12. T. Mora.*An Introduction to Commutative and Noncommutative Gröbner Bases.* Theoretical Computer Science, N. 134, (1994) 131-173.
13. A. Storjohann. *Near optimal algorithms for computing Smith normal forms of integer matrices.*In Lakshman, Y. N. ed., ISSAC'96: Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation, July 24-26, 1996, Zurich, Switzerland, (1996) 267-274. New York, NY 10036, USA, ACM Press.
14. A. Storjohann. *Algorithms for Matrix Canonical Forms.* Ph.D. Thesis, Institut für Wissenschaftliches Rechnen, ETH-Zentrum, Zürich, Switzerland. (2000)
15. Terras, A. *Fourier analysis on finite groups and applications.* LMS Student Texts, 43. Cambridge University Press, Cambridge. (1999)
16. E. Teske. *A Space Efficient Algorithm for Group Structure Computation.* Math. Comput., V. 67, N. 224, (1998) 1637-1663.
17. E. Teske. *The Pohlig-Hellman Method Generalized for Group Structure Computation.* J. Symbolic Computation 27, Issue 6, (1999) 521-534.

Normal Forms and Integrability of ODE Systems

Alexander D. Bruno[1] and Victor F. Edneral[2]

[1] Keldysh Institute of Applied Mathematics, Moscow 25047 Russia
[2] Moscow State University, Moscow 119992 Russia
edneral@theory.sinp.msu.ru

Abstract. We consider a special case of the Euler–Poisson system describing the motion of a rigid body with a fixed point. It is the autonomous ODE system of sixth order with one parameter. Among the stationary points of the system we select two one-parameter families with resonance $(0, 0, \lambda, -\lambda, 2\lambda, -2\lambda)$ of eigenvalues of the matrix of the linear part. For the stationary points, we compute the resonant normal form of the system using a program based on the MATHEMATICA package. Our results show that in cases of the existence of an additional first integral of the system its normal form is degenerate. So we assume that the integrability of a system can be checked through its normal form.

1 Introduction

Let us consider an autonomous system of ordinary differential equations in a neighborhood of its stationary point. The matrix of the linear part of the system can be reduced to the Jordan form by a linear change of variables. After that, the nonlinear part of the system can be reduced to the resonant normal form by means of an invertible formal change of variables. The normal form has resonant terms only and can be reduced to a system of lesser order [1]. Until now the normal form was used to study stability of a stationary point in critical cases of zero and pure imaginary eigenvalues and to find periodic solutions and conditionally periodic solutions [3,5,7].

On the other hand, a lot of books and papers were devoted to integrable systems and to methods for distinguishing them. The first author noted that all normal forms of integrable systems are degenerated.

Here we study connection between normal forms and integrability of a system. For that, we compute normal forms of the Euler–Poisson equations [6], which describe the motion of a rigid body with a fixed point. It is the autonomous system of the sixth order. The first attempt to compute normal form for the Euler–Poisson system was made by Starzhinsky [9]. But without computer algebra tools, he was unable to compute enough account of terms. We use the program for analytical computation of the normal form [8]. This is a modification of the LISP based package NORT [7] for the MATHEMATICA system. Package NORT was created for the REDUCE system.

We consider a special case when the system has only one parameter and has a two-parameter family of stationary points. Among them we select two one-parameter families with eigenvalues $0, 0, \lambda, -\lambda, 2\lambda, -2\lambda$. For stationary points of the families, we compute the normal form of the system up to fifth order and see that it is degenerate in known integrable cases and it is not degenerate in known nonintegrable cases. We have found four values of the parameter of the system for which the normal form in one family is degenerate and in another family is nondegenerate. In these cases the system can have local integrability near one stationary point but the system is nonintegrable globally.

The paper consists of three Sections. In Sect. 1 we recall the definition of the normal form and some its properties. In Sect. 2 we recall the Euler–Poisson system, describe its special case, and select families of its stationary points. In Sect. 3 we compute some coefficients of normal forms, describe them in graphics and tables and discuss the results.

2 The Normal Form of a Nonlinear System

We recall the main aspects of the theory expounded in [1,2,3,4]. We consider the system of the order n

$$dX/dt \overset{\text{def}}{=} \dot{X} = AX + \varPhi(X), \quad X \overset{\text{def}}{=} (x_1, \dots, x_n), \tag{1.1}$$

in a neighborhood of the stationary point $X = 0$, supposing that the vector-function $\varPhi(X)$ is analytic at the point $X = 0$, and that its Taylor series does not contain constant and linear terms.

Let the linear substitution $X = TY$ transform the matrix A to its Jordan normal form $G = T^{-1}AT$, and the whole system (1.1) into the form

$$\dot{Y} = GY + \tilde{\varPhi}(Y). \tag{1.2}$$

Let the formal change of coordinates

$$Y = Z + B(Z), \tag{1.3}$$

where $B = (b_1, \dots, b_n)$ and $b_i(Z)$ are formal power series without constant and linear terms, transform system (1.2) into the system

$$\dot{Z} = GZ + \tilde{\varPsi}(Z) \overset{\text{def}}{=} \varPsi(Z). \tag{1.4}$$

We write it down in the form

$$\dot{u}_j = u_j g_j(Z) \overset{\text{def}}{=} u_j \sum g_{jQ} Z^Q, \quad Q + E_j \geq 0, \quad j = 1, \dots, n. \tag{1.5}$$

Since G is a Jordan matrix, then its diagonal $\varLambda = (\lambda_1, \dots, \lambda_n)$ consists of eigenvalues of the matrix A.

Definition 1. [1] *System (1.4), (1.5) is called the resonant normal form, if:*

a) G is a Jordan matrix;

b) in expansions (1.5) there are only resonant terms for which the scalar product

$$\langle Q, \Lambda \rangle \overset{\text{def}}{=} q_1\lambda_1 + \ldots + q_n\lambda_n = 0. \tag{1.6}$$

Theorem 1 (Theorem on the normal form[1,2,3]). *There exists a formal substitution* (1.3)*, which transforms System* (1.2) *into the normal form* (1.4)*,* (1.5)*,* (1.6)*.*

Property 1. If System (1.2) has a linear automorphism of the form $t, Y \to \delta t, \tilde{S}Y$, then the normal form (1.4)–(1.6) has the same linear automorphism $t, Z \to \delta t, \tilde{S}Z$.

Let us consider the case when $n = 6$, the matrix G is diagonal and

$$\lambda_1 = \lambda_2 = 0, \;\; \lambda_3 = -\lambda_4 \neq 0, \;\; \lambda_5 = -\lambda_6 \neq 0. \tag{1.7}$$

Then among integer solutions $Q = (q_1, \ldots, q_6)$ of (1.6), there are solutions with

$$q_3 = q_4, \;\; q_5 = q_6. \tag{1.8}$$

We will call resonant terms $g_{jQ}Z^Q$ in (1.5) with that property as **main ones**. If the number λ_3/λ_5 is not rational then the normal form (1.5) has the main resonant terms only. If the number λ_3/λ_5 is rational then the normal form (1.5) has also not main resonant terms.

If the system (1.5) has the linear automorphism

$$t, z_1, z_2, z_3, z_4, z_5, z_6 \to t, z_1, z_2, -z_3, -z_4, -z_5, -z_6, \tag{1.9}$$

then the sum $k \overset{\text{def}}{=} q_3 + q_4 + q_5 + q_6$ is even for all terms in the normal form (1.5). If $\lambda_5/\lambda_3 = 2$, then (1.6) is

$$q_3 - q_4 + 2(q_5 - q_6) = 0. \tag{1.10}$$

For $k \leq 4$, there are following admissible solutions (q_3, q_4, q_5, q_6) of (1.10) without property (1.8):

$$\begin{gathered}(2,0,-1,0),\;(0,2,0,-1) \text{ with } k=1,\\ (2,0,0,1),\;(0,2,1,0),\;(3,1,-1,0),\;(1,3,0,-1) \text{ with } k=3,\\ (-1,3,2,0),\;(3,-1,0,2),\;(4,0,-1,1),\;(0,4,1,-1) \text{ with } k=4.\end{gathered} \tag{1.11}$$

3 The Euler–Poisson Equations

Motions of a rigid body with a fixed point are described by the Euler–Poisson system of six equations [6]

$$\begin{aligned} A\dot{p} + (C-B)qr &= Mg(z_0\gamma_2 - y_0\gamma_3),\\ B\dot{q} + (A-C)pr &= Mg(x_0\gamma_3 - z_0\gamma_1),\\ C\dot{r} + (B-A)pq &= Mg(y_0\gamma_1 - x_0\gamma_2),\end{aligned} \tag{2.1}$$

$$\begin{aligned}\dot\gamma_1 &= r\gamma_2 - q\gamma_3,\\ \dot\gamma_2 &= p\gamma_3 - r\gamma_1,\\ \dot\gamma_3 &= q\gamma_1 - p\gamma_2,\end{aligned} \tag{2.2}$$

where $A, B, C, M, g, x_0, y_0, z_0$ are real constants, A, B, C are positive and satisfy the triangle inequalities. System (2.1), (2.2) has three first integrals

$$\begin{aligned}F_1 &\overset{\text{def}}{=} Ap^2 + Bq^2 + Cr^2 + 2Mg(x_0\gamma_1 + y_0\gamma_2 + z_0\gamma_3) = \text{const},\\ F_2 &\overset{\text{def}}{=} Ap\gamma_1 + Bq\gamma_2 + Cr\gamma_3 = \text{const},\\ F_3 &\overset{\text{def}}{=} \gamma_1^2 + \gamma_2^2 + \gamma_3^2 = \text{const}.\end{aligned} \tag{2.3}$$

We will consider system (2.1), (2.2) with

$$A = B, \ \ Mgx_0/B = -1, \ \ y_0 = z_0 = 0. \tag{2.4}$$

Let us introduce the parameter $c = C/B$. Then system (2.1) takes the form

$$\begin{aligned}\dot p &= (1-c)qr,\\ \dot q &= (c-1)pr - \gamma_3,\\ \dot r &= \gamma_2/c.\end{aligned} \tag{2.5}$$

It has the single parameter $c \in (0, 2]$. System (2.5), (2.2) has an additional first integral $F_4 = \text{const}$ in two following cases.

If $c = 1$ (the Lagrange–Poisson case), $F_4 = p = \text{const}$. In that case there exists one more additional analytic first integral $F_5 = \text{const}$ [6].

If $c = 1/2$ (the S. Kovalevskaya case), $F_4 = (p^2 - q^2 + 2\gamma_1)^2 + (2pq + 2\gamma_2)^2 = \text{const}$ and System (2.5), (2.2) is integrable in quadratures, but it has no fifth analytic first integral. If was proven that system (2.5), (2.2) has no additional analytic first integral for other values $c \in (0, 2]$ [10].

System (2.5), (2.2) has the linear automorphism

$$t, p, q, r, \gamma_1, \gamma_2, \gamma_3 \to t, p, -q, -r, \gamma_1, -\gamma_2, -\gamma_3. \tag{2.6}$$

System (2.5), (2.2) has the two-parameter family of stationary points

$$\begin{aligned}&p = p_0 = \text{const}, \ \ q = q_0 = 0, \ \ r = r_0 = 0,\\ &\gamma_1 = \gamma_1^0 = 1, \ \ \gamma_2 = \gamma_2^0 = 0, \ \ \gamma_3 = \gamma_3^0 = 0.\end{aligned} \tag{2.7}$$

Near each stationary point (2.7) we introduce the local coordinates

$$P = p - p_0, \ \ q, r, \ \ \varGamma = \gamma_1 - 1, \gamma_2, \gamma_3. \tag{2.8}$$

System (2.2), (2.5) in the local coordinates (2.8) is

$$\begin{aligned}\dot P &= (1-c)qr,\\ \dot q &= (c-1)p_0 r - \gamma_3 + (c-1)Pr,\\ \dot r &= \gamma_2/c,\\ \dot\varGamma &= r\gamma_2 - q\gamma_3,\\ \dot\gamma_2 &= -r + p_0\gamma_3 + P\gamma_3 - r\varGamma,\\ \dot\gamma_3 &= q - p_0\gamma_2 + q\varGamma - P\gamma_3.\end{aligned} \tag{2.9}$$

The characteristic equation of the matrix of linear part of system (2.9) is

$$\lambda^6 + \alpha\lambda^4 + \beta\lambda^2 = 0, \tag{2.10}$$

where

$$\alpha = p_0^2 + 1 + \frac{1}{c}, \quad \beta = \frac{1}{c} + p_0^2\left(\frac{1}{c} - 1\right). \tag{2.11}$$

Equation (2.10) has two zero roots $\lambda_1 = \lambda_2 = 0$ and twin roots $\lambda_3 = -\lambda_4, \ \ \lambda_5 = -\lambda_6$. Now in family (2.7) we want to select points where $2\lambda_3 = \lambda_5$, i.e., $4\lambda_3^2 = \lambda_5^2$. From (2.10) we see that $2\lambda^2 = -\alpha \pm \sqrt{\alpha^2 - 4\beta}$. Let

$$2\lambda_3^2 = -\alpha + \sqrt{\alpha^2 - 4\beta}, \quad 2\lambda_5^2 = -\alpha - \sqrt{\alpha^2 - 4\beta}. \tag{2.12}$$

Equality $\lambda_5^2 = 4\lambda_3^2$ means that

$$-\alpha - \sqrt{\alpha^2 - 4\beta} = -4\alpha + 4\sqrt{\alpha^2 - 4\beta}. \tag{2.13}$$

Cancelling similar terms, squaring and cancelling again, we obtain from (2.13)

$$25\beta = 4\alpha^2. \tag{2.14}$$

Hence from (2.12) we have

$$2\lambda_3^2 = -\alpha + \frac{3}{5}\alpha = -\frac{2}{5}\alpha, \quad 2\lambda_5^2 = -\frac{8}{5}\alpha$$

and

$$\lambda_3^2 = -\alpha/5, \quad \lambda_5^2 = -4\alpha/5. \tag{2.15}$$

According to (2.11), equality (2.14) selects points with

$$4\left(p_0^2 + 1 + \frac{1}{c}\right)^2 = 25\left[\frac{1}{c} + p_0^2\left(\frac{1}{c} - 1\right)\right],$$

i.e.

$$p_0^2 = \frac{17 - 33c + 5\delta_2\sqrt{9 - 34c + 41c^2}}{8c} \overset{\text{def}}{=} p_0^2(\delta_2, c), \tag{2.16}$$

where $\delta_2 = \pm 1$. Again according to (2.11) and (2.16)

$$\alpha = 5\frac{5(1-c) + \delta_2\sqrt{9 - 34c + 41c^2}}{8c}. \tag{2.17}$$

Thus, from the two-parameter family (2.8) we have selected four one-parameter families

$$\mathcal{F}(\delta_1, \delta_2, c) : \ p_0 = \delta_1\sqrt{p_0^2(\delta_2, c)}, \quad \delta_1, \delta_2 = \pm 1, \quad c \in (0, 2] \tag{2.18}$$

with the resonance $\lambda_5 = 2\lambda_3$. Families (2.18) are intersected only at points where $p_0 = 0$, because the polynomial $9 - 34c + 41c^2$ has no real root. Equation $p_0 = 0$, i.e. $(17-33c)^2 = 25(9-34c+41c^2)$ according to (2.16), has only one root lesser than 2

$$c = c_1 \overset{\text{def}}{=} 1/4 = 0.25. \tag{2.19}$$

Equality $p_0 = 0$ for $c = 1/4$ takes place only for $\delta_2 = -1$. According to (2.15) the eigenvalue $\lambda_3 = 0$, if $\alpha = 0$, i.e. $25(1-c)^2 = 9 - 34c + 41c^2$ due to (2.17). That equation has the single positive root

$$c = c_2 \overset{\text{def}}{=} (\sqrt{5}-1)/2 \simeq 0.618034. \tag{2.20}$$

Equality $\alpha = 0$ for $c = c_2$ takes place again only for $\delta_2 = -1$. Hence families (2.18) with $\delta_2 = 1$ have no singular values of c, and with $\delta_2 = -1$ have two singular values (2.19) and (2.20). In what follows we will consider only families (2.18) with $\delta_1 = 1$, i.e. only two families

$$\mathcal{F}(1,1,c) \ \text{ and } \ \mathcal{F}(1,-1,c), \ \ c \in (0,2]. \tag{2.21}$$

On the family $\mathcal{F}(1,1,c)$ the value p_0 is real, and the eigenvalue λ_3 is purely imaginary. On the family $\mathcal{F}(1,-1,c)$ the value p_0 is real for $c < 1/4$ and is purely imaginary for $c > 1/4$, and the eigenvalue λ_3 is purely imaginary for $c \in (0, c_2)$ and is real for $c \in (c_2, 2]$.

4 Computation and Analysis of Normal Forms

Near stationary points of families (2.21) we computed normal forms (1.5) of system (2.9) up to terms of some order $m \overset{\text{def}}{=} q_1 + q_2 + \ldots + q_6$. For that, we used the program [8]. All calculations carried out in rational arithmetic and floating point numbers are approximations of exact results in this paper.

Due to automorphism (2.6) and to property 1 of Section 1, the normal forms (1.5) have the automorphism (1.9) and the sum $k \overset{\text{def}}{=} q_3 + q_4 + q_5 + q_6$ is even for all its terms. We considered sums

$$g_3(Z) + g_4(Z) \ \text{ and } \ g_5(Z) + g_6(Z). \tag{3.1}$$

For the normal form (1.5), it appears that for $m = q_1 + q_2 + k = 4$

$$\hat{g}_3 + \hat{g}_4 = a\frac{z_4^3 z_5^2}{z_3} - a\frac{z_3^2 z_6^2}{z_4}, \ \ \hat{g}_5 + \hat{g}_6 = b\frac{z_3^4 z_6}{z_5} - b\frac{z_4^4 z_5}{z_6}, \tag{3.2}$$

and all other terms cancel. Terms in (3.2) have power exponents (1.11) and their coefficients $a(c)$, $b(c)$ depend of δ_2 and c. It appears that for $\delta_2 = 1$ both coefficients a and b are purely imaginary, but for $\delta_2 = -1$ they both are purely imaginary if $c \in (0, c_2)$ and are real if $c \in (c_2, 2]$.

For $\delta_2 = 1$ plots of functions $\operatorname{Im} a(c)$ and $\operatorname{Im} b(c)$ are shown in Fig. 1. Four vertical lines in the Figs. correspond to $c = 0.25$, $c = 0.5$, $c = c_2$ and $c = 1$. All zeros of functions a and b are common and are situated at three points c:

$$c_3 \overset{\text{def}}{=} 0.252778, \ \ 0.5, \ \ 1. \tag{3.3}$$

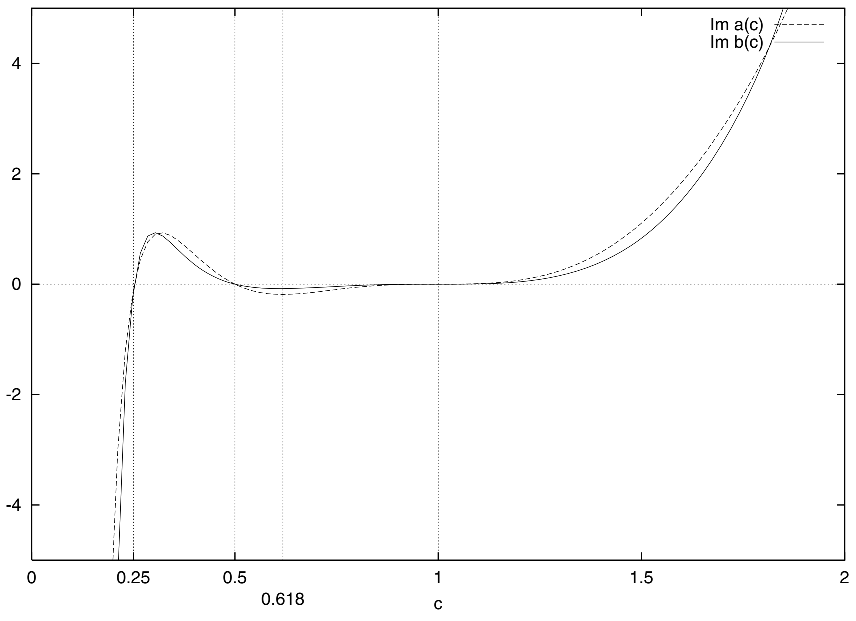


Fig. 1. The coefficients $\mathrm{Im}\, a(c), \mathrm{Im}\, b(c)$ for $\delta_2 = 1$

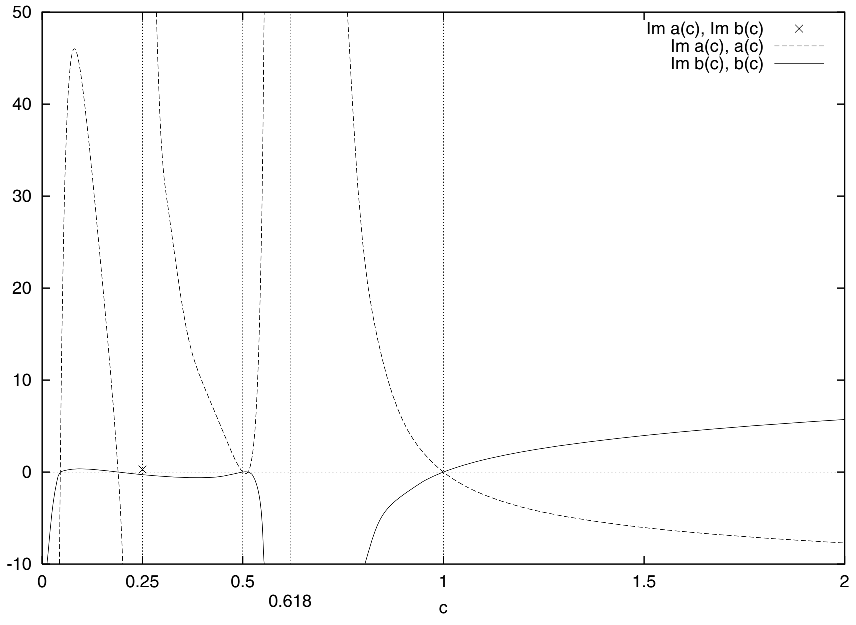


Fig. 2. The coefficients $\mathrm{Im}\, a(c), \mathrm{Im}\, b(c)$ and $a(c), b(c)$ for $\delta_2 = -1$

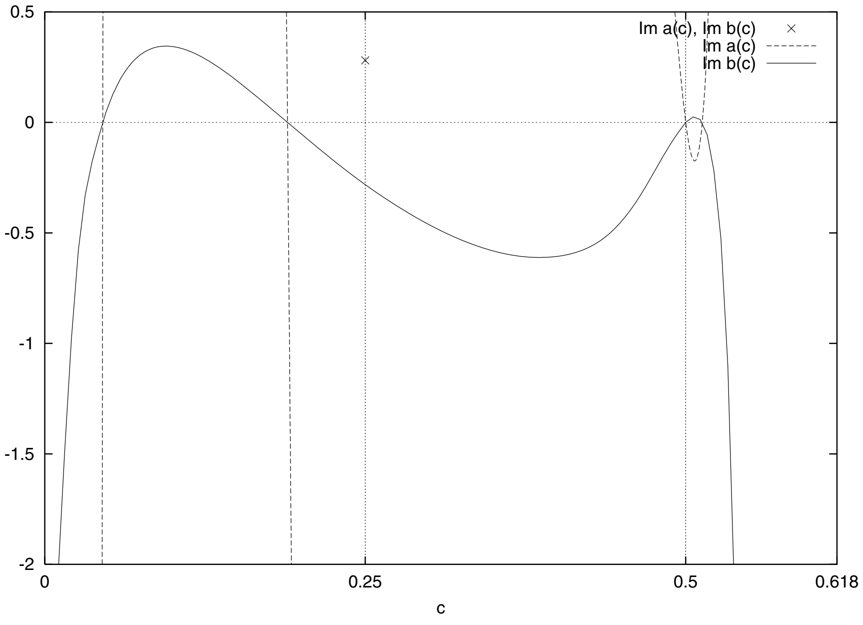


Fig. 3. The coefficients $\operatorname{Im} a(c), \operatorname{Im} b(c)$ for $\delta_2 = -1$ in a detailed scale

Functions $\operatorname{Im} a(c)$ and $\operatorname{Im} b(c)$ have two finite extrema each and near $c = 0$ their asymptotics are $-0.75c^{-2.5}$ and $-0.4c^{-3.5}$.

For $\delta_2 = -1$ Figs. 2 and 3 show plots of functions $\operatorname{Im} a(c)$ and $\operatorname{Im} b(c)$ for $c \in (0, c_2)$ and of functions $a(c)$ and $b(c)$ for $c \in (c_2, 2]$. Fig. 3 has bigger scale to better show small values of the functions. Near the singular point c_2 these functions tend to infinity as $10^3(c-c_2)^{-4}$ and as $-10^3(c-c_2)^{-4}$. Near the singular point $c_1 = 1/4$ the function $\operatorname{Im} a(c)$ tends to infinity as hyperbola $1.45/(c-0.25)$ but $\operatorname{Im} a(1/4) = 0.28125\ldots$ and the function $\operatorname{Im} b(c)$ tends to value $-0.28125\ldots$, but $\operatorname{Im} b(1/4) = \operatorname{Im} a(1/4) = 0.28125\ldots$ All zeros of functions $a(c)$ and $b(c)$ are common and are situated at five points c:

$$c_4 \overset{\text{def}}{=} 0.04522, \quad c_5 \overset{\text{def}}{=} 0.189372, \quad 0.5, \quad c_6 \overset{\text{def}}{=} 0.512902, \quad 1. \tag{3.4}$$

Functions $\operatorname{Im} a(c)$ and $\operatorname{Im} b(c)$ have five finite extrema together and near $c = 0$ their asymptotics are $-1.5c^{-1.5}$ and $-0.2c^{-0.5}$.

In $c = 1$ for $\delta_2 = \pm 1$ we computed normal forms up to order $m = 8$. Sums (3.1) for them are zeros. We assume that they are zeros in all orders m and that it is a consequence of the existence of five independent first integrals of system (2.9).

In $c = 1/2$ for $\delta_2 = \pm 1$ we computed normal forms up to order $m = 6$. It appears that sums (3.1) are

$$\begin{aligned}
&\hat{g}_3 + \hat{g}_4 = \xi_1(z_3^4 z_6^2 - z_4^4 z_5^2) + (\xi_2 z_1 + \xi_3 z_2 + \\
&+\xi_4 z_1^2 + \xi_5 z_1 z_2 + \xi_6 z_2^2)\left(\frac{z_4^3 z_5^2}{z_3} - \frac{z_3^3 z_6^2}{z_4}\right) + \xi_7\left(\frac{z_4^3 z_5^3 z_6}{z_3} - \frac{z_3^3 z_5 z_6^3}{z_4}\right), \\
&\hat{g}_5 + \hat{g}_6 = \eta_1(z_3^4 z_6^2 - z_4^4 z_5^2) + (\eta_2 z_1 + \eta_3 z_2 + \\
&+\eta_4 z_1^2 + \eta_5 z_1 z_2 + \eta_6 z_2^2)\left(\frac{z_3^4 z_6}{z_5} - \frac{z_4^4 z_5}{z_6}\right) + \eta_7\left(\frac{z_3^5 z_4 z_6}{z_5} - \frac{z_3 z_4^5 z_5}{z_6}\right),
\end{aligned} \tag{3.5}$$

where values of coefficients ξ_l/i and η_l/i are in Table 1, i.e. all sums do not vanish. So we assumed that vanishing sums (3.2) in order $m = 4$ is a consequence of the existence of one additional analytic first integral.

Table 1.

δ_2	λ_3	l	1	2	3	4	5	6	7
1	$-i$	ξ_l/i	−7.31	−0.914	1.29	1.69	−0.266	−2.55	−1.37
		η_l/i	−0.328	−0.219	0.309	0.569	0.490	−0.336	1.75
−1	$\frac{-i}{2}$	ξ_l/i	10.4	21.3	−32.1	58.5	−313.	−328.	−27.7
		η_l/i	0	0	0	1.14	2.30	6.10	0

The zero points c_j, $j = 3, 4, 5, 6$ of the functions $a(\delta_2, c)$ are known only approximately. So to study in them sums (3.1) in the order $m = 6$, we replaced each point c_j by two close points c_{j1}, c_{j2} with properties

$$c_{j1} < c_j < c_{j2}, \quad \operatorname{Im} a(\delta_2, c_{j1}) \cdot \operatorname{Im} a(\delta_2, c_{j2}) < 0, \quad j = 3, 4, 5, 6,$$

where $\delta_2 = 1$, if $j = 3$, and $\delta_2 = -1$, if $j = 4, 5, 6$. At these eight points c_{jk} we computed the normal form up to the order $m = 6$. Values of coefficients ξ_l and η_l in sums (3.5) at each pair of points c_{j1}, c_{j2} coincide in first two digits. Hence we conclude that at points c_3, c_4, c_5, c_6 the normal forms are nondegenerate in the order $m = 6$. It is possible that in some neighborhoods of the points $\mathcal{F}(1, 1, c_3)$, $\mathcal{F}(1, -1, c_4)$, $\mathcal{F}(1, -1, c_5)$ and $\mathcal{F}(1, -1, c_6)$ system (2.9) has local additional analytic first integrals. But for $c = c_3, c_4, c_5, c_6$ system (2.9) has no global additional analytic first integrals [10] and may be that near the points $\mathcal{F}(1, -1, c_3)$, $\mathcal{F}(1, 1, c_4)$, $\mathcal{F}(1, 1, c_5)$ and $\mathcal{F}(1, 1, c_6)$ there are no local additional integrals.

5 Conclusion

If the considered system has an additional global first integral then its normal form is degenerate for both families of stationary points. In the contrary cases the normal form has not such a degeneracy. It is either nondegenerate for both families or degenerate for one family only. There are four cases of such partial degeneracy. We assume that in these four cases the system has local first integrals near the places of degeneracy of its normal form.

At the moment we have also produced the analysis as above for resonances other than 1:2.

Acknowledgements. The paper was supported by RFBR, grant 05-01-00050, and by grant of President RF Sc. Sch.-1450.2003.2.

References

1. Bruno, A.D.: The normal form of differential equations. Doklady Akad. Nauk SSSR **157** (1964) 1276–1279, in Russian, Soviet Math. Doklady **5** (1964) 1105–1108, in English
2. Bruno, A.D.: Analytical form of differential equations. Trudy Mosk. Mat. Obshch. **25** (1971) 119–262; **26** (1972) 199–239, in Russian, Trans. Moscow Math. Soc. **25** (1971) 131–288; **26** (1972) 199–239, in English
3. Bruno, A.D.: Local Methods in Nonlinear Differential Equations. Springer, Berlin (1989)
4. Bruno, A.D.: Normal forms. Mathematics and Computers in Simulation **45**, nos. 5–6 (1998) 413–427
5. Bruno, A.D.: Power Geometry in Algebraic and Differential Equations. Elsevier Science B. V. (2000)
6. Golubev, V.V.: Lectures on Integration of Equations of Motion of a Rigid Body Around a Fixed Point. Moscow, GITTL (1953), in Russian
7. Edneral, V.F.: A symbolic approximation of periodic solutions of the Henon–Heiles system by the normal form method. Mathematics and Computers in Simulation **45**, nos. 5–6 (1998) 445–463
8. Edneral, V.F., Khanin, R.: Application of the resonant normal form to high order nonlinear ODEs using MATHEMATICA. Nuclear Instruments and Methods in Physics Research A **502**, nos. 2–3 (2003) 643–645
9. Starzhinsky, V.M.: Applied Methods in Nonlinear Oscillation. Nauka, Moscow (1977), in Russian. Ch. IX
10. Ziglin, S.L.: Branching solutions and nonexistence of integrals in the Hamiltonian mechanics I, II. Functional Analysis and its Applications **16**, no. 3 (1982) 30–41; **17**, no. 1 (1983) 8–23

Resultant-Based Methods for Plane Curves Intersection Problems

Laurent Busé[1], Houssam Khalil[2], and Bernard Mourrain[1]

[1] INRIA, Galaad, 2004 Route des Lucioles,
BP 93, 06902 Sophia Antipolis Cedex, France
{lbuse, mourrain}@sophia.inria.fr
[2] Laboratoire de Mathématiques Appliquées de Lyon, UCBL and CNRS,
22 Avenue Claude Bernard, 69622 Villeurbanne Cedex, France
khalil@maply.univ-lyon1.fr

Abstract. We present an algorithm for solving polynomial equations, which uses generalized eigenvalues and eigenvectors of resultant matrices. We give special attention to the case of two bivariate polynomials and the Sylvester or Bezout resultant constructions. We propose a new method to treat multiple roots, detail its numerical aspects and describe experiments on tangential problems, which show the efficiency of the approach. An industrial application of the method is presented at the end of the paper. It consists in recovering cylinders from a large cloud of points and requires intensive resolution of polynomial equations.

1 Introduction

We present an algorithm, which uses generalized eigenvalues and eigenvectors, for solving systems of two bivariate polynomials $p(x, y) = q(x, y) = 0$ in $\mathbb{R}$, with $p, q \in \mathbb{R}[x, y]$. Such a problem can be viewed as computing the intersection points of two implicitly defined plane algebraic curves, which is a key operation in Computer Aided Geometric Design. Several methods already exist to solve this problem, and we refer the interested reader to [7] for a general overview. The one that we present is based on a matrix formulation and the use of generalized eigenvalues and eigenvectors of two companion matrices built from the Sylvester or Bezout matrix of p and q, following methods initiated by Stetter (see [6, chapters 2] for a nice overview), which allows us to apply a preprocessing step and to develop efficient solvers for specific applications. A new improvment that we propose is the treatment of multiple roots, although these roots are usually considered as obstacles for numerical linear algebra techniques; the numerical difficulties are handled with the help of singular value decomposition (SVD) of the matrix of eigenvectors associated to each eigenvalue.

Similar methods, based on eigen-computations, have already been addressed, in particular in the papers [13,4,3]. Both methods project the roots of the system $p(x, y) = q(x, y) = 0$, say, on the x-coordinates, using a matrix formulation, and then lift them up, as well as their multiplicity. We follow the same path, but improve these results. Indeed, in [3] two multiplication maps have to be computed to project the roots on the x-coordinates, whereas we only need one such map; in [13], the lifting of the x-coordinates of the roots is done by solving simultaneously two univariate polynomials, with approximate coefficients, obtained by substitutting the x-coordinnates by an approximation of an eigenvalue. This procedure is numerically very delicate. In our approach, we gather both the projection and the lifting step into a single eigen-problem and propose a stable way to treat multiple roots.

The paper is organized as follows. In section 2, we recall the needed tools. In section 3, we briefly give an overview of Bezout's theorem and explain how to use generalized eigenvalues and eigenvectors for solving two bivariate polynomial systems. Then, in section 4 we describe the numerical problems, how we remedy to them by using SVD, and give the algorithm. Finally, in section 5, we give some examples and show an industrial application.

2 Preliminaries

In this section, we first recall how a univariate polynomial can be solved via eigenvalue computations. The algorithm we provide in this paper can be seen as a generalization of this simple but important result. Then we briefly survey the very basic properties of the well-known Sylvester resultant. Finally we recall some elementary definitions of generalized eigenvalues and eigenvectors.

2.1 A Univariate Polynomial Solver

Let $\mathbb{K}$ be any field and $f(x) := f_d x^d + f_{d-1}x^{d-1} + \cdots + f_1 x + f_0 \in \mathbb{K}[x]$. Using the standard Euclidean polynomial division it is easy to see that the quotient algebra $\mathcal{A} = \mathbb{K}[x]/I$, where I denotes the principal ideal of $\mathbb{K}[x]$ generated by the polynomial $f(x)$, is a vector space over $\mathbb{K}$ of dimension d with canonical basis $\{1, x, \ldots, x^{d-1}\}$.

Consider $M_x : \mathcal{A} \to \mathcal{A}$, the multiplication by x in $\mathcal{A}$. It is straightforward to check that the matrix of M_x in the basis $\{1, x, \ldots, x^{d-1}\}$ is given by

$$M_x = \begin{bmatrix} 0 \cdots 0 & -f_0/f_d \\ 1 \ddots & \vdots \\ \vdots \quad 0 & \vdots \\ 0 \quad\quad 1 & -f_{d-1}/f_d \end{bmatrix},$$

(the column on the far right corresponds to the Euclidean division of x^d by f). The characteristic polynomial of M_x equals $\frac{(-1)^d}{f_d} f(x)$ and the roots of the polynomial f can be recovered, with their corresponding multiplicities, by computing the eigenvalues of the multiplication map M_x. See e.g. [12].

2.2 The Sylvester Resultant

Let A be a commutative ring, which is assumed to be a domain, and suppose we are given two polynomials in $A[x]$ of respective degree d_0 and d_1

$$f_0(x) := c_{0,0} + c_{0,1}x + \cdots + c_{0,d_0}x^{d_0},$$
$$f_1(x) := c_{1,0} + c_{1,1}x + \cdots + c_{1,d_1}x^{d_1}.$$

The Sylvester matrix of f_0 and f_1 (in degree (d_0, d_1)) is the matrix whose columns contain successively the coefficients of the polynomials $f_0, xf_0, \ldots, x^{d_1-1}f_0, f_1, xf_1, \ldots, x^{d_0-1}f_1$ expanded in the monomial basis $\{1, x, \ldots, x^{d_0+d_1-1}\}$. More precisely, it is the matrix of the A-linear map

$$\begin{aligned} \sigma : A[x]_{<d_1} \oplus A[x]_{<d_0} &\to A[x]_{<d_0+d_1} \\ (q_0, q_1) &\mapsto f_0q_0 + f_1q_1 \end{aligned} \tag{1}$$

in the monomial bases $\{1, \ldots, x^{d_1-1}\}, \{1, \ldots, x^{d_0-1}\}$ and $\{1, \ldots, x^{d_0+d_1-1}\}$, where $A[x]_{<k}$ denotes the set of polynomials in $A[x]$ of degree lower or equal to $k-1$. It is thus a square matrix of size $d_0 + d_1$ which has the following form:

$$S := \begin{array}{c} \overbrace{\begin{array}{cccc} f_0 \cdots x^{d_1-1}f_0 & & f_1 \cdots x^{d_0-1}f_1 & \end{array}}^{d_0+d_1} \\ \left(\begin{array}{cc|cc} c_{0,0} & 0 & c_{1,0} & 0 \\ \vdots \ \ddots & & \vdots \ \ddots & \\ & c_{0,0} & & c_{1,0} \\ c_{0,d_0} & \vdots & c_{1,d_1} & \vdots \\ \ddots & \vdots & \ddots & \vdots \\ 0 & c_{0,d_0} & 0 & c_{1,d_1} \end{array}\right) \end{array} \begin{array}{l} \\ 1 \\ x \\ \vdots \\ \vdots \\ \vdots \\ x^{d_0+d_1-1} \end{array}$$

Proposition 1. *If A is a field, then we have* $\dim \ker(S) = \deg(\gcd(f_0, f_1))$.

Proof. Let $d(x) := \gcd(f_0, f_1)$. We easily check that the kernel of (1) is the set of pairs $\left(r(x)\frac{f_1(x)}{d(x)}, -r(x)\frac{f_0(x)}{d(x)}\right)$ where $r(x)$ is an arbitrary polynomial in $A[x]_{<\deg(d)}$. See also [11, corollary 5.3] for another description of this kernel.

Definition 1. *The resultant of the polynomials f_0 and f_1, denoted $Res(f_0, f_1)$, is the determinant of the Sylvester matrix of f_0 and f_1.*

The resultant $\mathrm{Res}(f_0, f_1)$ is thus an element in A. It has been widely studied in the literature, as it provides a way to eliminate the variable x from the polynomial system $f_0(x) = f_1(x) = 0$. Indeed,

Proposition 2. $\mathrm{Res}(f_0, f_1) = 0$ *if and only if either $c_{0,d_0} = c_{1,d_1} = 0$, or either $f_0(x)$ and $f_1(x)$ have a common root in the algebraic closure of the quotient field* $\mathrm{Frac}(A)$ *of A (or equivalently f_0 and f_1 are not coprime in* $\mathrm{Frac}(A)[x]$*).*

Proof. See e.g. [12, IV §8].

Another matrix, called the Bezout matrix, can be used to compute $\mathrm{Res}(f_0, f_1)$. We now suppose, without loss of generality, that $d_1 \geq d_0$.

Definition 2. *The Bezoutian of the polynomials f_0 and f_1 is the element in $A[x, y]$ defined by*

$$\Theta_{f_0,f_1}(x, y) := \frac{f_0(x)f_1(y) - f_1(x)f_0(y)}{x - y} = \sum_{i,j=0}^{d_1-1} \theta_{i,j}x^iy^j,$$

and the Bezout matrix is $B_{f_0,f_1} := (\theta_{i,j})_{0 \leq i,j \leq d_1-1}$.

The matrix B_{f_0,f_1} is a $d_1 \times d_1$-matrix which is symmetric. It is thus a smaller matrix than the Sylvester matrix, but has more complicated entries, and its determinant still equals the resultant of f_0 and f_1 (up to a power of c_{1,d_1}):

Proposition 3 ([11] §5.4). *We have*

$$\det(B_{f_0,f_1}) = (-1)^{\frac{1}{2}d_1(d_1-1)}\,(c_{1,d_1})^{d_1-d_0}\,\mathrm{Res}(f_0,f_1).$$

Moreover, if A is a field then $\dim\ker(B_{f_0,f_1}) = \deg(\gcd(f_0,f_1))$.

2.3 Generalized Eigenvalues and Eigenvectors

Let A and B be two matrices of size $n \times n$. A *generalized eigenvalue* of A and B is a value in the set

$$\lambda(A,B) := \{\lambda \in \mathbb{C} : \det(A - \lambda B) = 0\}.$$

A vector $x \neq 0$ is called a *generalized eigenvector* associated to the eigenvalue $\lambda \in \lambda(A,B)$ if $Ax = \lambda Bx$. The matrices A and B have n generalized eigenvalues if and only if $\mathrm{rank}(B) = n$. If $\mathrm{rank}(B) < n$, then $\lambda(A,B)$ can be finite, empty, or infinite. Note that if $0 \neq \mu \in \lambda(A,B)$ then $1/\mu \in \lambda(B,A)$. Moreover, if B is invertible then $\lambda(A,B) = \lambda(B^{-1}A, I) = \lambda(B^{-1}A)$, which is the ordinary spectrum of $B^{-1}A$.

Recall that an $n \times n$ matrix $T(x)$ with polynomial entries can be equivalently written as a polynomial with $n \times n$ matrix coefficients. If $d = \max_{i,j}\{\deg(T_{ij}(x))\}$, we obtain $T(x) = T_d x^d + T_{d-1}x^{d-1} + \cdots + T_0$, where T_i are $n \times n$ matrices.

Definition 3. *The companion matrices of $T(x)$ are the two matrices A, B given by*

$$A = \begin{pmatrix} 0 & I & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & I \\ T_0^t & T_1^t & \cdots & T_{d-1}^t \end{pmatrix},\; B = \begin{pmatrix} I & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & I & 0 \\ 0 & \cdots & 0 & -T_d^t \end{pmatrix}.$$

And we have the following interesting property:

Proposition 4. *With the above notation, the following equivalence holds:*

$$T^t(x)v = 0 \Leftrightarrow (A - xB)\begin{pmatrix} v \\ xv \\ \vdots \\ x^{d-1}v \end{pmatrix} = 0.$$

Proof. A straightforward computation.

3 The Intersection of Two Plane Algebraic Curves

From now on we assume that $\mathbb{K}$ is an algebraically closed field and that $p(x,y)$ and $q(x,y)$ are two polynomials in $\mathbb{K}[x,y]$. Our aim will be to study and compute their common roots. This problem can be interpreted geometrically. Polynomials p and q define two algebraic curves in the affine plane $\mathbb{A}^2$ (having coordinates (x,y)), and we would like to know their intersection points.

Hereafter we will consider systems $p(x,y) = q(x,y) = 0$ having only a *finite* number of roots. This condition is not quite restrictive since it is sufficient (and necessary) to require that polynomials p and q are coprime in $\mathbb{K}[x,y]$ (otherwise we divide them by their gcd).

In the case where one of the curve is a line, i.e. one of the polynomial has degree 1, the problem is reduced to solving a univariate polynomial. Indeed, one may assume e.g. that $q(x,y) = y$ and thus we are looking for the roots of $p(x,0) = 0$. Let us denote them by $z_1,\dots,z_s$. Then we know that

$$p(x,0) = c(x-z_1)^{\mu_1}(x-z_2)^{\mu_2}\cdots(x-z_s)^{\mu_s},$$

where the z_i's are assumed to be distinct and c is a non-zero constant in $\mathbb{K}$. The integer μ_i, for all $i = 1,\dots,s$, is called the *multiplicity* of the root z_i, and it turns out that $\sum_{i=1}^{s}\mu_i = \deg(p)$ if $p(x,y)$ does not vanish at infinity in the direction of the y-axis (in other words, if the homogeneous part of p of highest degree does not vanish when $y = 0$). This later condition can be avoid in the projective setting: let $p^h(x,y,t)$ be the homogeneous polynomial obtained by homogenizing $p(x,y)$ with the new variable t, then we have

$$p(x,0,t) = c(x-z_1t)^{\mu_1}(x-z_2t)^{\mu_2}\cdots(x-z_st)^{\mu_s}t^{\mu_\infty},$$

where μ_∞ is an integer corresponding to the multiplicity of the root at infinity, and $\mu_\infty + \sum_{i=1}^{s}\mu_i = \deg(p)$. Moreover, it turns out that the roots $z_1,\dots,z_s$ and their corresponding multiplicities can be computed by eigenvalues and eigenvectors computation (see section 2.1).

In the following we generalize this approach to the case in which p and q are bivariate polynomials of arbitrary degree. For this we first need to recall the notion of multiplicity in this context. Then we will show how to recover the roots from multiplication maps.

3.1 Intersection Multiplicity

Let $p(x,y), q(x,y)$ be two coprime polynomials in $\mathbb{K}[x,y]$, $\mathcal{C}_p$ and $\mathcal{C}_q$ the corresponding algebraic plane curves, $I := (p,q)$ the ideal they generate in $\mathbb{K}[x,y]$ and $\mathcal{A} := \mathbb{K}[x,y]/I$ the associated quotient ring. We denote by $z_1 = (x_1,y_1),\dots,z_s = (x_s,y_s)$ the distinct intersection points in $\mathbb{A}^2$ of $\mathcal{C}_p$ and $\mathcal{C}_q$ (i.e. the distinct roots of the system $p(x,y) = q(x,y) = 0$).

A modern definition of the intersection multiplicity of $\mathcal{C}_p$ and $\mathcal{C}_q$ at a point z_i is (see [8, §1.6])

$$i(z_i, \mathcal{C}_p \cap \mathcal{C}_q) := \dim_{\mathbb{K}} \mathcal{A}_{z_i} < \infty,$$

where the point $z_i \in \mathbb{A}^2$ is here abusively (but usually) identified with its corresponding prime ideal $(x - x_i, y - y_i)$ in $\mathbb{K}[x, y]$, $\mathcal{A}_{z_i}$ denoting the ordinary localization of the ring $\mathcal{A}$ by this prime ideal. As a result, the finite $\mathbb{K}$-algebra $\mathcal{A}$ (which is actually finite if and only if $p(x, y)$ and $q(x, y)$ are coprime in $\mathbb{K}[x, y]$) can be decomposed as the direct sum

$$\mathcal{A} = \mathcal{A}_{z_1} \oplus \mathcal{A}_{z_2} \oplus \cdots \oplus \mathcal{A}_{z_s}$$

and consequently $\dim_{\mathbb{K}} \mathcal{A} = \sum_{i=1}^{s} i(z_i, \mathcal{C}_p \cap \mathcal{C}_q)$.

The intersection multiplicities can be computed using a resultant. The main idea is to project "algebraically" the intersection points on the x-axis. To do this let us see both polynomials p and q in $A[y]$ where $A := \mathbb{K}[x]$, that is to say as univariate polynomials in y with coefficients in the ring A which is a domain; we can rewrite

$$p(x, y) = \sum_{i=0}^{d_1} a_i(x) y^i, \quad q(x, y) = \sum_{i=0}^{d_2} b_i(x) y^i. \tag{2}$$

Their resultant (with respect to y) is an element in A which is non-zero, since p and q are assumed to be coprime in $A[y]$, and which can be factorized, assuming that $a_{d_1}(x)$ and $b_{d_2}(x)$ do not vanish simultaneously, as (see proposition 2)

$$\mathrm{Res}(p, q) = c \prod_{i=1}^{r} (x - \alpha_i)^{\beta_i} \tag{3}$$

where the α_i's are distinct elements in $\mathbb{K}$ and $\{\alpha_1, \ldots, \alpha_r\} = \{x_1, \ldots, x_s\}$ (as sets). For instance, if all the $x_i's$ are distinct then we have $r = s$ and $\alpha_i = x_i$ for all $i = 1, \ldots, s$. Moreover, we have (see e.g. [8, §1.6]):

Proposition 5. *For all $i \in \{1, \ldots, r\}$, the integer β_i equals the sum of all the intersection multiplicities of the points $z_j = (x_j, y_j) \in \mathcal{C}_p \cap \mathcal{C}_q$ such that $x_j = \alpha_i$:*

$$\beta_i = \sum_{z_j = (x_j, y_j) | x_j = \alpha_i} i(z_j, \mathcal{C}_p \cap \mathcal{C}_q).$$

As a corollary, if all the x_i's are distinct (this can be easily obtained by a linear change of coordinates (x, y)) then $i(z_i, \mathcal{C}_p \cap \mathcal{C}_q)$ is nothing but the valuation of $\mathrm{Res}(p, q)$ at $x = x_i$ (i.e. the exponent of the largest power of $(x - x_i)$ which divides $\mathrm{Res}(p, q)$). Another corollary of this result is the well-known Bezout theorem for algebraic plane curves, which is better stated in the projective context. Let us denote by $p^h(x, y, t)$ and $q^h(x, y, t)$ the homogeneous polynomials in $\mathbb{K}[x, y, t]$ obtained from p and q by homogenization with the new variable t.

Proposition 6 (Bezout theorem). *If p and q are coprime then the algebraic projective plane curves associated to $p^h(x, y, t)$ and $q^h(x, y, t)$ intersect in $\deg(p) \deg(q)$ points in $\mathbb{P}^2$, counted with multiplicities*

Proof. It follows directly from the previous proposition and the well-known fact that $\mathrm{Res}(p^h, q^h)$ is a homogeneous polynomial in $\mathbb{K}[x, t]$ of degree $\deg(p) \deg(q)$ (see e.g. [12], [17]).

3.2 Intersection Points

We take again the notation of (2) and (3). We denote by $S(x)$ the Sylvester matrix of $p(x,y)$ and $q(x,y)$ seen in $A[y]$ (assuming that both has degree at least one in y); thus $\det(S(x)) = \mathrm{Res}(p,q)$. From (3), we deduce immediately that $\det(S(x))$ vanishes at a point $x_0 \in \mathbb{K}$ if and only if either there exists y_0 such that $p(x_0,y_0) = q(x_0,y_0) = 0$ or $a_{d_1}(x_0) = b_{d_2}(x_0) = 0$ (in which case the intersection point is at infinity). Therefore one may ask the following question: *given a point x_0 such that $\det(S(x_0)) = 0$, how may we recover all the possible points y_0 such that $p(x_0,y_0) = q(x_0,y_0) = 0$?*

Suppose we are given such a point x_0, and assume that $a_{d_1}(x_0)$ and $b_{d_2}(x_0)$ are not both zero. If $\ker(S(x_0)^t)$ is of dimension one, then it is easy to check that there is only one point y_0 such that $p(x_0,y_0) = q(x_0,y_0) = 0$, and moreover any element in $\ker(S(x_0)^t)$ is a multiple of the vector $[1, y_0, \cdots, y_0^{d_1+d_2-1}]$. Therefore to compute y_0, we only need to compute a basis of $\ker(S(x_0)^t)$ and to compute the quotient of its second coordinate by its first one. In case of multiple roots, this construction can be generalized as follows:

Proposition 7. *With the above notation, let $\Lambda_1, \cdots, \Lambda_d$ be any basis of the kernel $\ker(S(x_0)^t)$, $\mathbf{\Lambda}$ be the matrix whose i^{th} row is the vector Λ_i, Δ_0 be the $d \times d$-submatrix of $\mathbf{\Lambda}$ corresponding to the first d columns and Δ_1 be the $d \times d$-submatrix of $\mathbf{\Lambda}$ corresponding to the columns of index $2, 3, \ldots, d+1$. Then $\lambda(\Delta_1, \Delta_0)$ is the set of roots of the equations $p(x_0,y) = q(x_0,y) = 0$ (i.e. the set of y-coordinates of the intersection points of $\mathcal{C}_p$ and $\mathcal{C}_q$ above $x = x_0$).*

Proof. Let $g(y) := \gcd(p(x_0,y), q(x_0,y)) \in \mathbb{K}[y]$ and $k_1 = \deg(p(x_0,y)), k_2 = \deg(q(x_0,y))$. By proposition 1, we know that $d = \deg(g(y))$. Consider the map

$$\begin{aligned} \mathrm{rem}_{x_0} : \mathbb{K}[y]_{<k_1+k_2} &\to \mathbb{K}[y]_{<d} \\ t(y) &\mapsto \mathrm{remainder}(t(y), g(y)), \end{aligned}$$

which sends $t(y)$ on the remainder of its Euclidean division by $g(y)$. Observe that, for all $i = 0, \ldots, k_1 + k_2 - 2$, $\mathrm{rem}_{x_0}(y^{i+1}) = \mathrm{rem}_{x_0}(y\, \mathrm{rem}_{x_0}(y^i))$. Consequently, $\mathrm{rem}_{x_0}(y^{i+1}) = \mathtt{M}_y \mathrm{rem}_{x_0}(y^i)$, where $\mathtt{M}_y$ is the operator of multiplication by y in the quotient ring $\mathbb{K}[y]/(g(y))$ (see section 2.1). Let Δ be the matrix of rem_{x_0} in the monomial basis of $\mathbb{K}[y]_{<k_1+k_2}$, Δ_0 a submatrix of Δ of size $d \times k$ and Δ_1 the submatrix obtained by shifting the index of columns of Δ_0 in Δ by 1. Then by the previous remark, we have $\Delta_1 = \mathtt{M}_y \Delta_0$. In particular, if we choose the first d columns of Δ, we have $\Delta_0 = \mathrm{Id}$ so that $\lambda(\Delta_0, \Delta_1)$ are the roots of $g(y) = p(x_0,y) = q(x_0,y) = 0$.

Now, $S(x_0)$ is the matrix in the monomial bases of the map (1), denoted here σ_{x_0}. By construction $\mathrm{rem}_{x_0} \circ \sigma_{x_0} = 0$ and $\dim\ker(\sigma_{x_0}) = \deg(g) = \dim \mathrm{Im}(\mathrm{rem}_{x_0})$, which shows that the rows of the matrix of rem_{x_0} form a basis of $\ker(S(x_0)^t) \equiv \mathbb{K}[y]_{<d}$. Since by any change of basis of $\ker(S(x_0)^t)$ the equality $\mathtt{M}_y \Delta_0 = \Delta_1$ remains true, the claim is proved.

Remark 1. In this theorem, observe that the construction of Δ_0 and Δ_1 is always possible, that is to say that $\mathbf{\Lambda}$ has at least $d+1$ columns, because we assumed that both polynomials p and q depend on the variable y which implies that the number of rows in the matrix $S(x_0)$ is always strictly greater than $\deg(\gcd(p(x_0,y),q(x_0,y)))$.

The previous theorem shows how to recover the y-coordinates of the roots of the system $p = q = 0$ from the Sylvester matrix. However, instead of the Sylvester matrix we can use the Bezout matrix to compute $\mathrm{Res}(p,q) \in A[x]$ (see section 2). This matrix has all the required properties to replace the Sylvester matrix in all the previous results, except in proposition 7. Indeed, the Bezout matrix being smaller than the Sylvester matrix, the condition given in remark 1 is not always fulfilled; using the Bezout matrix in proposition 7 requires that for all roots x_0 of $\mathrm{Res}(p,q)$,

$$\max(\deg(p(x_0,y),\deg(q(x_0,y)) > \deg(\gcd(p(x_0,y),q(x_0,y)).$$

This condition may fail: take for instance $x_0 = -1$ in the system

$$\begin{cases} p(x,y) = x^2y^2 - 2y^2 + xy - y + x + 1 \\ q(x,y) = y + xy \end{cases}$$

Note that the use of the Bezout matrix gives, in practice, a faster method because it is a smaller matrix than the Sylvester matrix, even if its computation takes more time (the savings in the eigen-computations is greater).

3.3 The Algorithm

We are now ready to describe our resultant-based solver. According to proposition 4, we replace the computation of the resultant, its zeroes, and the kernel of $S^t(x_0)$ (or the Bezout matrix) for each root x_0 by the computation of generalized eigenvalues and eigenvectors of the associated companion matrices A and B. This computation can be achieved with the QZ algorithm [9].

Algorithm 1. EXACT ARITHMETIC VERSION

1. *Compute the Bezout matrix $B(x)$ of p and q.*
2. *Compute the associated companion matrices A and B (see proposition 4).*
3. *Compute the generalized eigenvalues and eigenvectors of (A,B). The eigenvalues give the x-coordinates of the intersection points, and their multiplicities give the sum of the intersection multiplicities of the points above them (see proposition 5). The eigenvector spaces give bases of* $\ker(B(x_0))$ *in proposition 7; their dimensions give the degree of the gcd of p and q at these points.*
4. *For each value x_0,*
   (a) *If the number of associated eigenvectors is at least* $\max(\deg(p(x_0,y)),$ $\deg(q(x_0,y)))$*, which is the size of $B(x_0)$, compute Δ_0 and Δ_1 by using a basis of* $\ker(S(x_0)^t)$*,*
   (b) *if not, then compute Δ_0 and Δ_1 by using the eigenvectors associated to x_0.*
5. *Compute the eigenvalues of (Δ_1,Δ_0) which give the y-coordinates of the intersection points above x_0 (see proposition 7).*

4 Numerical Difficulties

The following difficulties are found in numerical computations:

a. In order to compute the x-coordinates of intersection points, it is necessary to compute the real generalized eigenvalues of (A, B). Numerically, some of these values can contain a nonzero imaginary part. Generally such a problem is encountered for multiple eigenvalues.
b. What do we mean by a numerical multiple x-coordinate?
c. How do we choose the linearly independent vectors among the computed eigenvectors?

In order to solve these problems, we need the singular value decomposition (SVD for short).

4.1 SVD as Remedy

Let us recall known results on the singular value decomposition.

Theorem 1 ([9]). *For a real matrix A of size $m \times n$, there exist two orthogonal matrices*

$$U = [u_1, \cdots, u_m] \in \mathbb{R}^{m\times m}, \text{ and } V = [v_1, \cdots, v_n] \in \mathbb{R}^{n\times n}$$

such that

$$U^t A V = diag(\sigma_1, \cdots, \sigma_p) \in \mathbb{R}^{m\times n}, \quad p = \min\{m, n\}, \text{ and}$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0.$$

σ_i are called the ith singular value of A, u_i and v_i are respectively the ith left and right singular vectors.

Definition 4. *For a matrix A, the numerical rank of A is defined by:*

$$\text{rank}(A, \epsilon) = \min\{\text{rank}(B) : \left\| \frac{A}{\|A\|} - B \right\| \leq \epsilon\},$$

where $\|\cdot\|$ denotes either $\|\cdot\|_2$, the spectral norm, or $\|\cdot\|_F$, the Frobenius norm.

Theorem 2. *([9]) If $A \in \mathbb{R}^{m\times n}$ is of rank r and $k < r$ then*

$$\sigma_{k+1} = \min_{\text{rank}(B)=k} \|A - B\|.$$

Theorem 2 shows that the smallest singular value is the distance between A and all the matrices of rank $< p = \min\{m, n\}$. Thus if $r_\epsilon = \text{rank}(A, \epsilon)$ then

$$\sigma_1 \geq \cdots \geq \sigma_{r_\epsilon} > \epsilon\sigma_1 \geq \sigma_{r_{\epsilon+1}} \geq \cdots \geq \sigma_p.$$

As a result, in order to determine the rank of a matrix A, we find the singular values $(\sigma_i)_i$ so that

$$\text{rank}(A, \epsilon) = \max\{i : \frac{\sigma_i}{\sigma_1} > \epsilon\}.$$

For difficulty "a" we choose a small $\epsilon \in \mathbb{R}$ and consider as real any eigenvalue whose imaginary part is of absolute value less than ϵ.

For difficulty "b", we choose possibly a different small $\epsilon \in \mathbb{R}$ and gather the points which lay in an interval of size ϵ. According to the following proposition, the proposed algorithm remains effective:

Proposition 8. *Assume that $\{\xi_1 \cdots \xi_n\} \subseteq \lambda(A, B)$ with corresponding generalized eigenvectors $v_1^{(1)}, \cdots, v_{k_1}^{(1)}, \cdots, v_1^{(n)}, \cdots, v_{k_n}^{(n)}$ such that $\sum_{i=1}^n k_i < d_1 + d_2$. We define $\mathbf{\Lambda}$ to be the matrix whose rows are $v_1^{(1)}, \cdots, v_{k_1}^{(1)}, \cdots, v_1^{(n)}, \cdots, v_{k_n}^{(n)}$ and Δ_0 (resp. Δ_1) to be its first (resp. second) left $(k_1 + \cdots + k_n) \times (k_1 + \cdots + k_n)$ block. Then, the generalized eigenvalues of Δ_1 and Δ_0 give the set of y-coordinates of the intersection points above $\xi_1 \cdots \xi_n$; in other words, clusters of x-coordinates are thus gathered and regarded as only one point whose multiplicity equals the sum of the multiplicities of the gathered points.*

Proof. We define $\mathbf{\Lambda}_i$ as the matrix corresponding to eigenvectors of ξ_i, it is a matrix of size $k_i \times (d_1 + d_2)$, then $\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 \\ \vdots \\ \mathbf{\Lambda}_n \end{bmatrix}$. For $\mathbf{\Lambda_i}$, we take $\Delta_0^{(i)}$ the first left $k_i \times (k_1 + \cdots + k_n)$ block, and $\Delta_1^{(i)}$ the second block. Then, according to the proof of proposition 7, $\Delta_1^{(i)} = M_{y,\xi_i} \Delta_0^{(i)}$, where $M_{y,\xi}$ is the matrix of multiplication by y in the quotient ring $\mathbb{K}[y]/\gcd(p(\xi_i, y), q(\xi_i, y))$. It follows that $\Delta_1 = M_y \Delta_0$ where

$$M_y = \begin{pmatrix} M_{y,\xi_1} & & 0 \\ & \ddots & \\ 0 & & M_{y,\xi_n} \end{pmatrix}.$$

Moreover, Δ_0 is invertible and thus $\lambda(\Delta_1, \Delta_0) = \lambda(M_y) = \{\lambda(M_{y,\xi_i}), i = 1 \cdots n\}$.

For difficulties "c", it is necessary to use the singular value decomposition: If x a generalized eigenvalue of A and B, and E the matrix of the associated eigenvectors, then the singular value decomposition of E is

$$E = U \Sigma V^t, \quad \text{with } \Sigma = \begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix}$$

and r is the rank of E. Let

$$E' = E \cdot V = U \cdot \Sigma = [\sigma_1 u_1 \cdots \sigma_r u_r \, 0 \cdots 0] = [e'_1 \cdots e'_r \, 0 \cdots 0].$$

The matrix E' is also a matrix of eigenvectors, because the changes are done on the columns, and V is invertible. Hence E' and E have the same rank r, and

$E'' = [e'_1 \cdots e'_r]$ is a matrix of linearly independent eigenvectors. Consequently E'' can be used to build Δ_0 and Δ_1.

We now give the complete algorithm which can be used in the presence of singular points:

Algorithm 2. NUMERICAL APPROXIMATION OF THE ROOTS OF A BIVARIATE POLYNOMIAL SYSTEM.

1. *Compute the Bezout matrix $B(x)$ of p and q, and compute the matrices A and B.*
2. *Compute the generalized eigenvalues and eigenvectors of (A, B).*
3. *Eliminate the imaginary eigenvalues and the values at infinity, and gather the close points (with the input ϵ).*
4. *For each of all the close points represented by ξ, take the matrix E of the associated eigenvectors.If their number $\geq d_1 + d_2$ then E gives a basis of* $\ker S^t(\xi)$
5. *Do a singular value decomposition of E.*
6. *Calculate the rank of E by testing the σ_i until $\sigma_{r+1} < \sigma_r \epsilon$ is found (the rank is thus determined to be r).*
7. *Let E'' be the r first columns of $E' = E \cdot V$. Define Δ_0 as the first $r \times r$ block in E'', and Δ_1 as the second $r \times r$ block in E''.*
8. *Compute the generalized eigenvalues of (Δ_1, Δ_0).*

Remark 2. On step 3, there are very often eigenvalues at infinity, because the size of A and B is larger than the degree of the resultant. Therefore, generally one obtains more eigenvalues than roots, some of the eigenvalues being at infinity.

5 Numerical Experiments and Application

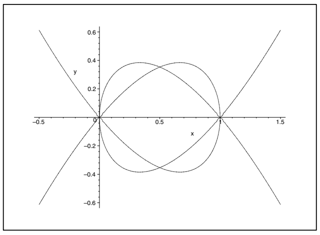
Our algorithm[1] is implemented in C++, in the library C++ called "SYNAPS" (SYmbolic and Numeric ApplicationS)[2]. It uses the linear algebra library "LAPACK" (in FORTRAN), for approximate computation of eigenvalues and eigenvectors. The following computations were done on a Pentium4, 3.06 Ghz computer.

5.1 Examples

Example 1: $\begin{cases} p = y^2 - x^2 + x^3 \\ q = y^2 - x^3 + 2x^2 - x \end{cases}$

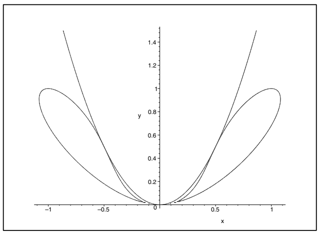
[1] `http://www-sop.inria.fr/galaad/logiciels/synaps/html/bivariate__res_8H-source.html`

[2] `http://www-sop.inria.fr/galaad/logiciels/synaps/`



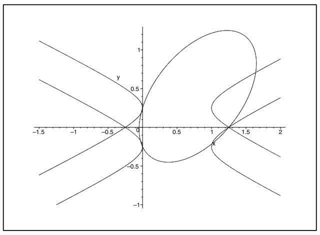
```
The intersection points are: (epsilon=10^-6).
(x1, y1) = (0,0)           of multiplicity 2
(x2, y2) = (0.5, -0.35)    of multiplicity 1
(x3, y3) = (0.5,0.35)      of multiplicity 1
(x4, y4) = (1, -8.2e-25)   of multiplicity 2
Execution time = 0.004s
```

Example 2: $\begin{cases} p = x^4 - 2x^2y + y^2 + y^4 - y^3 \\ q = y - 2x^2 \end{cases}$



```
The intersection points are: (epsilon=10^-6).
(x1, y1) = (1.6e-09,0)     of multiplicity 4
(x2, y2) = (-0.5,0.5)      of multiplicity 2
(x3, y3) = (0.5,0.5)       of multiplicity 2
Execution time = 0.005s
```

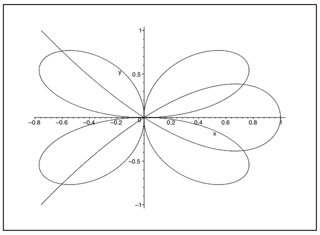
Example 3: $\begin{cases} p = 400y^4 - 160y^2x^2 + 16x^4 + 160y^2x - 32x^3 - 50y^2 + 6x^2 \\ \qquad +10x + \frac{25}{16} \\ q = y^2 - x + x^2 - \frac{5}{6} - \frac{1}{6} \end{cases}$



```
the intersection points are: (epsilon=10^-6).
(x1, y1) = (1.4e-16, -0.25) of multiplicity 1
(x2, y2) = (1.4e-16,0.25)   of multiplicity 1
```

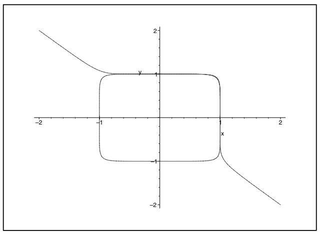
```
(x3, y3) = (-0.03, -0.16)   of multiplicity 1
(x4, y4) = (-0.01,0.18)     of multiplicity 1
(x5, y5) = (1.64,0.70)      of multiplicity 1
(x6, y6) = (1, -0.21)       of multiplicity 1
(x7, y7) = (1.25,1.5e-08)   of multiplicity 2
Execution time = 0.007s
```

Example 4: $\begin{cases} p = x^6 + 3x^4y^2 + 3x^2y^4 + y^6 - 4x^2y^2 \\ q = y^2 - x^2 + x^3 \end{cases}$



```
the intersection points are: (epsilon=10^-3).
(x1, y1) = (-3e-16,2e-17)  of multiplicity 8
(x2, y2) = (-0.60,-0.76)   of multiplicity 1
(x3, y3) = (-0.60,0.76)    of multiplicity 1
(x4, y4) = (0.72, -0.37)   of multiplicity 1
(x5, y5) = (0.72,0.37)     of multiplicity 1
Execution time = 0.011s
```

Example 5: $\begin{cases} p = x^9 + y^9 - 1 \\ q = x^{10} + y^{10} - 1 \end{cases}$



```
The intersection points are: (epsilon=2.10^-2).
(x1, y1) = (-0.01,1)      of multiplicity 9
(x2, y2) = (1,0)          of multiplicity 9
Execution time = 0.111s
```

5.2 Table of Comparison

We give here a table[3] comparing our method (called GEB below), with two other algorithms implemented in SYNAPS for solving the curve intersection

[3] See http://www-sop.inria.fr/galaad/data/curve2d/

problem. The first uses Sturm-Habicht sequences [10], the second uses normal form computations (the "Newmac" method) [14], [15].

Ex.	Degree	N.s	Execution time			M
			GEB	Sturm	Newmac	
ex001	3,3	4	0.004	0.002	0.029	1.1e-16
ex002	4,2	3	0.005	0.005	0.026	8.8e-16
ex003	4,2	7	0.007	0.010	0.029	6.8e-10
ex004	6,3	5	0.011	0.005	0.041	1.7e-15
ex005	9,10	2	0.111	0.130	1.705	6.6e-15
ex006	6,4	4	0.013	0.010	0.063	1.8e-10
ex007	8,7	41	0.089	0.211	0.306	1.7e-05
ex008	8,6	27	0.085	0.091	0.283	2.1e-05
ex009	6,4	2	0.006	0.003	0.032	3.2e-11
ex010	4,3	5	0.005	0.003	0.033	2.5e-13
ex011	4,3	3	0.004	0.003	0.030	7.1e-15
ex012	5,4	2	0.008	0.010	0.053	4.7e-14
ex013	6,5	5	0.021	0.02	0.058	1.7e-11
ex014	11,10	5	0.083	0.194	0.332	4.4e-06
ex015	6,5	4	0.015	0.01	0.049	2.6e-15
ex016	8,7	41	0.088	0.208	0.305	1.7e-05

(N.s denotes the number of solutions and M denotes the number $\max(\max_i |p(x_i,y_i)|, \max_i |q(x_i,y_i)|)$ where (x_i,y_i) are the computed solutions.

5.3 Cylinders Passing Through Five Points

An important problem in CAD modeling is the extraction of a set of geometric primitives properly describing a given 3D point cloud obtained by scanning a real scene. If the extraction of planes is considered as a well-solved problem, the extraction of circular cylinders, these geometric primitives are basically used to represent "pipes" in an industrial environment, is not easy and has been recently addressed. In this section, we describe an application of our algorithm to this problem which has been experimented by Thomas Chaperon from the MENSI[4] company.

In [2], the extraction of circular cylinders was done in two steps: after the computation of estimated unit normals of the given 3D set of points, their Gaussian images give the possible cylinders as great circles on the Gaussian sphere. In order to avoid the estimation of the unit normals another approach was presented in [1]. First, being given 5 points randomly selected in our 3D point cloud, the author gave a polynomial system whose roots correspond to the cylinders passing through them (recall that 5 is the minimum number of points defining generically a finite number of cylinders, actually 6 in the *complex* numbers). Then the method consists in finding these cylinders, actually their direction, for almost all set of 5 points in the whole point could, and extract the "clusters of

[4] `http://www.mensi.fr/`

directions" as a primitive cylinder. In [5], motivated by another application in metrology, the authors carefully studied the problem of finding cylinders passing through 5 given points in the space; in particular they produce two polynomials in three homogeneous variables whose roots corresponds to the expected cylinders. In the sequel, using the same polynomial system that we will briefly recall, we show that our new solver can speed-up the solving step.

Modelisation. We briefly recall from [5] how the problem of finding the 6 complex cylinders passing through 5 (sufficiently generic) points can be translated into a polynomial system. We actually only seek the direction of these cylinders since their axis and radius follow then easily (see e.g. [2][appendix A]). We will denote by p_1, p_2, p_3, p_4, p_5 the five given points defining six cylinders. We first look for the cylinders passing only through the first four points that we can assume, w.l.o.g., to be $p_1 = (0,0,0), p_2 = (x_2,0,0), p_3 = (x_3,y_3,0)$ and $p_4 = (x_4,y_4,z_4)$. We also denote by $\mathbf{t} = (l,m,n)$, where $l^2+m^2+n^2 = 1$, the unitary vector identifying a direction in the 3D space; note that $\mathbf{t}$ also identified with the projective point $(l:m:n)$ in $\mathbb{P}^2$.

Let π be the plane passing through the origin and orthogonal to $\mathbf{t}$ and let (X,Y,Z) be a coordinate system such that the two first axes are in π and the third one has direction $\mathbf{t}$. Among all the coordinate changes sending (x,y,z) onto (X,Y,Z) we choose the one given by the following matrix, where $\rho = m^2+n^2$:

$$\begin{pmatrix} \rho & -\frac{lm}{\rho} & -\frac{nl}{\rho} \\ 0 & \frac{n}{\rho} & -\frac{m}{\rho} \\ l & m & n \end{pmatrix}.$$

The orthogonal projection q_i of p_i on π has coordinates, in the system (X,Y,Z) :

$$(X_i, Y_i, Z_i) := \left(\rho x_i - \frac{lm}{\rho} y_i - \frac{nl}{\rho} z_i, \frac{n}{\rho} y_i - \frac{m}{\rho} z_i, 0 \right).$$

The points p_1, p_2, p_3, p_4 belong to a cylinder of direction $\mathbf{t}$ if and only if the points q_1, q_2, q_3, q_4 are cocyclic in π, i.e. if and only if

$$\begin{vmatrix} 1 & 1 & 1 & 1 \\ X_1 & X_2 & X_3 & X_4 \\ Y_1 & Y_2 & Y_3 & Y_4 \\ X_1^2+Y_1^2 & X_2^2+Y_2^2 & X_3^2+Y_3^3 & X_4^2+Y_4^2 \end{vmatrix} = 0.$$

We denote this previous determinant by $\mathcal{C}_{p_1,p_2,p_3,p_4}(l,m,n)$. The coordinates of q_1 and q_2 satisfy $X_1 = Y_1 = 0$, $X_2 = \rho x_2$, and $Y_2 = 0$. Moreover, for $i = 3, 4$ we have

$$X_i^2 + Y_i^2 = |q_i|^2 = (\mathbf{t}.\mathbf{t})|p_i|^2 - (t.p_i)^2.$$

Therefore, by developing the determinant $\mathcal{C}_{p_1,p_2,p_3,p_4}(l,m,n)$ one can show that it equals (see [5] for more details)

$$x_2^2(m^2+n^2) \begin{vmatrix} l & x_3 & x_4 \\ m & y_3 & y_4 \\ n & 0 & z_4 \end{vmatrix} - x_2 \begin{vmatrix} m & y_3 & y_4 \\ n & 0 & z_4 \\ 0 & (\mathbf{t}.\mathbf{t})|p_3|^2 - (t.p_3)^2 & (\mathbf{t}.\mathbf{t})|p_4|^2 - (t.p_4)^2 \end{vmatrix}.$$

Seeing $\mathbf{t}$ as a projective point in $\mathbb{P}^2$, this equation thus defines an algebraic curve of degree 3 in $\mathbb{P}^2$.

We deduce that a cylinder of direction $\mathbf{t} = (l, m, n)$ goes through the 5 points p_1, p_2, p_3, p_4, p_5 if (a necessary condition)

$$\mathcal{C}_{p_1,p_2,p_3,p_4}(l, m, n) = 0 \text{ and } \mathcal{C}_{p_1,p_2,p_3,p_5}(l, m, n) = 0. \tag{4}$$

These two equations define two algebraic curves in $\mathbb{P}^2$ and hence intersect into exactly 9 points (counted with multiplicity) from the Bezout theorem. But the directions $\boldsymbol{p_1p_2}$, $\boldsymbol{p_1p_3}$ and $\boldsymbol{p_2p_3}$ are solutions of (4) but correspond to a cylinder if and only if they are of multiplicity at least 2. Consequently the system (4) gives us the 6 solutions we are looking for and three extraneous solutions that we know by advance and that we can easily eliminate after the resolution of (4).

For instance, there are exactly 6 *real* cylinders passing through the 5 following points:

$$\begin{array}{lll}
x_1 := 0, & y_1 := 0, & z_1 := 0, \\
x_2 := 1, & y_2 := 0, & z_2 := 0, \\
x_3 := \frac{1}{2}, & y_3 := \frac{\sqrt{3}}{2}, & z_3 := 0, \\
x_4 := \frac{1}{2}, & y_4 := \frac{1}{2\sqrt{3}}, & z_4 := \frac{\sqrt{2}}{\sqrt{3}}, \\
x_4 := \frac{1}{2}, & y_4 := \frac{1}{2\sqrt{3}}, & z_4 := -\frac{\sqrt{2}}{\sqrt{3}}.
\end{array}$$

By applying our algorithm we obtain the 6 directions:

```
(-0.81722,-1.8506e-17) of multiplicity 1
(-0.40929,-0.70721)    of multiplicity 1
(-0.40929,0.70721)     of multiplicity 1
(0.40929,-0.70721)     of multiplicity 1
(0.40929,0.70721)      of multiplicity 1
(0.81722,-1.8506e-17)  of multiplicity 1
```

Experimentations. We now turn to the experimentation of our method. We took 1000 sets of 5 random points, with integer coordinates between −100 and 100 and computed the 6 (complex) corresponding cylinders. Comparing the previously mentioned methods, we obtained the following timing:

method	GEB	Sturm	Newmac
time (sec.)	0.67	1.83	1.61

which shows a significant improvement, in the context of intensive computation.

6 Conclusion

In this paper, we presented a new algorithm for solving systems of two bivariate polynomial equations using generalized eigenvalues and eigenvectors of resultant matrices (both Sylvester and Bézout types). We proposed a new method to treat multiple roots, described experiments on tangential problems which show the efficiency of the approach and provided an industrial application consisting in recovering cylinders from a large cloud of points. We also performed a first numerical analysis of our approach; we plan to investigate it further following the ideas developed in [16], in order to control a posteriori the error on the roots.

References

1. T. Chaperon. A note on the construction of right circular cylinders through five 3d points. Technical report, Centre de Robotique, École des mines de Paris, january 2003.
2. T. Chaperon, F. Goulette, and C. Laurgeau. Extracting cylinders in full 3d data using a random sampling method and the gaussian image. In T. Ertl, B. Girod, G. Greiner, H. Nieman, and H. Seidel, editors, *Vision, Modeling and Visualization (VMV'01)*, Stuttgart, November 2001.
3. R. Corless, P. Gianni, and B. Trager. A reordered schur factorization method for zero-dimensional polynomial systems with multiple roots. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 133–140 (electronic), New York, 1997. ACM.
4. R. M. Corless, L. Gonzalez-Vega, I. Necula, and A. Shakoori. Topology determination of implicitly defined real algebraic plane curves. *An. Univ. Timişoara Ser. Mat.-Inform.*, 41(Special issue):83–96, 2003.
5. O. Devillers, B. Mourrain, F. P. Preparata, and P. Trebuchet. Circular cylinders through four or five points in space. *Discrete Comput. Geom.*, 29(1):83–104, 2003.
6. A. Dickenstein and I. Emiris, editors. *Solving Polynomial Equations: Foundations, Algorithms, and Applications*, volume 14 of *Algorithms and Computation in Mathematics*. Springer-Verlag, april 2005.
7. M. Elkadi and B. Mourrain. Symbolic-numeric tools for solving polynomial equations and applications. In A. Dickenstein and I. Emiris, editors, *Solving Polynomial Equations: Foundations, Algorithms, and Applications.*, volume 14 of *Algorithms and Computation in Mathematics*, pages 125–168. Springer, 2005.
8. W. Fulton. *Introduction to intersection theory in algebraic geometry*, volume 54 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC, 1984.
9. G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.
10. L. González-Vega and I. Necula. Efficient topology determination of implicitly defined algebraic plane curves. *Comput. Aided Geom. Design*, 19(9):719–743, 2002.
11. N. Kravitsky. Discriminant varieties and discriminant ideals for operator vessels in Banach space. *Integral Equations Operator Theory*, 23(4):441–458, 1995.
12. S. Lang. *Algebra*, volume 211 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, third edition, 2002.
13. D. Manocha and J. Demmel. Algorithms for intersecting parametric and algebriac curves ii; multiple intersections. *Computer vision, Graphics and image processing: Graphical models and image processing*, 57:81–100, 1995.
14. B. Mourrain. A new criterion for normal form algorithms. In *Applied algebra, algebraic algorithms and error-correcting codes (Honolulu, HI, 1999)*, volume 1719 of *Lecture Notes in Comput. Sci.*, pages 430–443. Springer, Berlin, 1999.

15. B. Mourrain and P. Trebuchet. Solving projective complete intersection faster. In *Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation (St. Andrews)*, pages 234–241 (electronic), New York, 2000. ACM.
16. S. Oishi. Fast enclosure of matrix eigenvalues and singular values via rounding mode controlled computation. *Linear Algebra Appl.*, 324(1-3):133–146, 2001. Special issue on linear algebra in self-validating methods.
17. B. L. Van der Waerden. *Modern algebra, Vol. II.* New-York, Frederick Ungar Publishing Co, 1948.

Symbolic Calculations in Studying the Stability of Dynamically Symmetric Satellite Motion

Carlo Cattani[1], Evgenii A. Grebenikov[2], and Alexander N. Prokopenya[3]

[1] DiFarma, University of Salerno,
Via Ponte Don Melillo, I-84084 Fisciano (SA), Italy
ccattani@unisa.it

[2] Computing Center of RAS, Vavilova str. 37,
117312 Moscow, Russia
greben@ccas.ru

[3] Brest State Technical University, Moskowskaya str. 267,
224017 Brest, Belarus
prokopenya@brest.by

Abstract. The stability of cylindrical precession of the dynamically symmetric satellite in the Newtonian gravitational field is studied. We consider the case when a center of mass of the satellite moves in an elliptic orbit, while the satellite rotates uniformly about the axis of its dynamical symmetry that is perpendicular to the orbit plane. In the case of the resonance 3 : 2 (Mercury type resonance) we have found the domains of instability of cylindrical precession of the satellite in the Liapunov sense and domains of its linear stability in the parameter space. Using the infinite determinant method we have calculated analytically the boundaries of the domains of instability as power series in the eccentricity of the orbit. All the calculations have been done with the computer algebra system *Mathematica*.

1 Introduction

It is well known that celestial mechanics and cosmic dynamics are just the fields where using computers for doing symbolic calculations turned out to be not only very productive but necessary as well [1]. Many problems in these fields require extremely bulky calculations for their solving. And the problem of motion stability is just a typical example. In the present paper we study the stability of stationary motion of dynamically symmetrical satellite in the Newtonian gravitational field. The case of cylindrical precession [2] is considered when a center of mass of the satellite moves in the elliptic orbit, while the satellite rotates uniformly about the axis of its dynamical symmetry that is perpendicular to the orbit plane. A particular case of the satellite motion without rotation was investigated in detail in [3]. Here we consider the resonance case $\beta = T_0/T_{rot} = 3/2$ (Mercury type resonance), where T_0 and T_{rot} are periods of the orbital motion and rotation, respectively. Note that the case of $\beta = 1$ is discussed shortly in [4].

It should be emphasized that the study of stability is the most complicated problem of qualitative theory of differential equations. Usually it starts from analysis of the linearized differential equations of the disturbed motion and determination of the domains of linear stability and instability of the system in the parameter space. Further analysis of the system stability in the Liapunov sense implies using both the classical stability theory of Liapunov–Poincaré and the methods of Kolmogorov–Arnold–Moser (KAM) theory [5]. The second approach is connected with construction of a sequence of canonical transformations, reducing the Hamiltonian of the system to the normal form, and application of Arnold–Moser and Markeev theorems [6]. This procedure implies rather bulky analytical calculations which can be reasonably done only with the help of modern computer algebra systems such as *Mathematica*, for example [7].

Here we focus on studying the stability of the satellite motion in linear approximation. Note that usually such problem is solved by means of calculating the characteristic exponents for the corresponding system of linear differential equations of the disturbed motion [8]. But in the considered case such calculations are so bulky that we can get the stability boundaries only in the first approximation in a small parameter [3], [4]. However, using the algorithm based on the infinite determinant method [9], [10], [11], we can quite easily find the boundaries between the domains of stability and instability in the parameter space in higher approximations. In the present paper we compute the stability boundaries with accuracy up to the sixth order in a small parameter. And although we consider here the resonance case $\beta = 3/2$, all the calculations may be easily re-done for any value of this parameter.

2 Equations of the Disturbed Motion

Let us consider the satellite as a rigid, dynamically symmetric body whose center of mass moves along an elliptic orbit in the central Newtonian gravitational field. We assume that its linear sizes are sufficiently small in comparison with the orbit size and, hence, the satellite motion about the center of mass is independent of its orbital motion [2]. In order to describe the satellite motion we introduce two coordinate systems: the system $Oxyz$ being rigidly attached to the satellite, with the Oz axis pointing along the satellite axis of dynamical symmetry, and the orbital system $OXYZ$ whose axes OZ and OY are directed along the radius vector of the satellite center of mass O and along the normal vector to the orbital plane, respectively. Orientation of the $Oxyz$ frame with respect to the orbital frame $OXYZ$ is specified by the Euler angles ψ, θ and φ. Then equations of motion of the satellite are just the canonical equations with the Hamiltonian function of the form [2]

$$\begin{aligned} H = \frac{1}{2(1+e\cos\nu)^2}\left(\frac{p_\psi^2}{\sin^2\theta} + p_\theta^2 + \alpha^2\beta^2(1-e^2)^3\cot^2\theta \right.\\ \left. -2\alpha\beta(1-e^2)^{3/2}p_\psi\frac{\cos\theta}{\sin^2\theta}\right) - p_\psi\cot\theta\cos\psi - p_\theta\sin\psi + \\ +\alpha\beta(1-e^2)^{3/2}\frac{\cos\psi}{\sin\theta} + \frac{3}{2}(\alpha-1)(1+e\cos\nu)\cos^2\theta \ , \end{aligned} \tag{1}$$

where p_ψ, p_θ are the momenta canonically conjugated to the angles ψ, θ, respectively; the satellite inertia parameter α is defined as the ratio of its polar and equatorial moments of inertia ($0 \le \alpha \le 2$); parameter β is equal to the ratio of periods of the satellite orbital motion T_0 and its proper rotation T with respect to an absolute frame of reference (note that T is just an integral of the motion); ν is the true anomaly, and e is an eccentricity of the satellite orbit.

It may easily be shown that the Hamiltonian equations of motion

$$\dot{\psi} = \frac{\partial H}{\partial p_\psi} \ , \ \dot{\theta} = \frac{\partial H}{\partial p_\theta} \ , \ \dot{p_\psi} = -\frac{\partial H}{\partial \psi} \ , \ \dot{p_\theta} = -\frac{\partial H}{\partial \theta} \ , \tag{2}$$

where a dot means a differentiation with respect to the true anomaly ν, have a solution

$$\psi = \pi \ , \ \theta = \frac{\pi}{2} \ , \ p_\psi = 0 \ , \ p_\theta = 0 \ . \tag{3}$$

It describes a stationary motion when the satellite rotates uniformly about its axis of dynamical symmetry that is perpendicular to the orbit plane. This kind of the satellite motion is known as the cylindrical precession [2].

In order to investigate the stability of cylindrical precession, we introduce small perturbations $q_1, \ q_2, \ p_1, \ p_2$ of the stationary solution (3), so that

$$\psi = \pi + q_1 \ , \ \theta = \frac{\pi}{2} + q_2 \ , \ p_\psi = p_1 \ , \ p_\theta = p_2 \ , \tag{4}$$

and expand the Hamiltonian function (1) in powers of $q_1, \ q_2, \ p_1, \ p_2$. Then (1) takes a form

$$H = H_2 + H_4 + \dots \ , \tag{5}$$

where

$$\begin{aligned} H_2 = {} & \frac{1}{2(1+e\cos\nu)^2}\Big(p_1^2 + p_2^2 + 2\alpha\beta(1-e^2)^{3/2} p_1 q_2 + \alpha^2\beta^2(1-e^2)^3 q_2^2\Big) \\ & + q_1 p_2 - q_2 p_1 + \frac{3}{2}(\alpha-1)(1+e\cos\nu) q_2^2 + \frac{1}{2}(1-e^2)^{3/2}\alpha\beta(q_1^2 - q_2^2) \ , \end{aligned} \tag{6}$$

$$\begin{aligned} H_4 = {} & \frac{1}{6(1+e\cos\nu)^2}\Big(3p_1^2 q_2^2 + 5(1-e^2)^{3/2}\alpha\beta p_1 q_2^3 + 2(1-e^2)^3\alpha^2\beta^2 q_2^4\Big) \\ & + \frac{1}{6}\left(3p_1 q_1^2 q_2 - p_2 q_1^3 - 2p_1 q_2^3 - 3(\alpha-1)(1+e\cos\nu) q_2^4\right) \\ & - \frac{1}{24}(1-e^2)^{3/2}\alpha\beta(q_1^4 - 6q_1^2 q_2^2 + 5q_2^4) \ . \end{aligned} \tag{7}$$

The linearized equations of the disturbed motion are easily obtained from (2) if we take into account only quadratic term (6) in the expansion (5). They can be written in the form

$$\dot{q_1} = \frac{1}{(1+e\cos\nu)^2}\Big(p_1 + (1-e^2)^{3/2}\alpha\beta q_2\Big) - q_2 \ ,$$

$$\dot{q_2} = q_1 + \frac{p_2}{(1+e\cos\nu)^2} \ , \ \dot{p_1} = -p_2 - (1-e^2)^{3/2}\alpha\beta q_1 \ ,$$
$$\dot{p_2} = \left(1 - \frac{(1-e^2)^{3/2}\alpha\beta}{(1+e\cos\nu)^2}\right)p_1 - \frac{(1-e^2)^3\alpha^2\beta^2}{(1+e\cos\nu)^2}\,q_2$$
$$+(1-e^2)^{3/2}\alpha\beta q_2 - 3(\alpha-1)(1+e\cos\nu)q_2 \ . \tag{8}$$

3 Determination of Linear Stability Domains

The canonical equations (8) form a system of four linear differential equations with periodic coefficients of the period 2π. Note that general properties of such systems have been studied quite well (see, for example, [8]). The right-hand sides in (8) are analytic functions of e at the point $e = 0$. Hence, the behaviour of solutions of (8) for sufficiently small e is determined by its characteristic exponents calculated for $e = 0$. And the system may be linearly stable only if for $e = 0$ all its characteristic exponents λ_j are different purely imaginary numbers which satisfy the following inequality:

$$\lambda_j \pm \lambda_k \neq iN \ (j,k = 1,2,3,4;\ N = 0,\ \pm1,\ \pm2,\ \ldots) \ , \tag{9}$$

where i is the imaginary unit ($i^2 = -1$).

In the case of $e = 0$ (8) is reduced to the system of four linear differential equations with constant coefficients. Its characteristic exponents are easily found and can be written in the form

$$\lambda_{1,2} = \pm i\sigma_1,\ \lambda_{3,4} = \pm i\sigma_2 \ , \tag{10}$$

where

$$\sigma_{1,2} = \frac{1}{\sqrt{2}}\Big(-1 + 3\alpha - 2\alpha\beta + \alpha^2\beta^2 \pm \big((-1 + 3\alpha - 2\alpha\beta + \alpha^2\beta^2)^2 -$$
$$-4(4 + \alpha^2\beta(3+\beta) - \alpha(3+5\beta))\big)^{1/2}\Big)^{1/2} \ . \tag{11}$$

Let us analyse the parameters $\sigma_{1,2}$ defined in (11) for $\beta = \frac{3}{2}$. If the inertia parameter α belongs to the interval

$$0 < \alpha < \frac{8}{9} \ , \tag{12}$$

then parameter σ_2 is a complex number. Hence, one of the characteristic exponents λ_3 or λ_4 has a positive real part, and the system (8) is unstable for $e = 0$. As the characteristic exponents are continuous functions of e then there will exist at least one characteristic exponent with a positive real part for sufficiently small $e > 0$ as well. According to Liapunov theorem on linearized stability [12], we can conclude that for $\beta = 3/2$ the solution (3) is unstable for sufficiently small value of the eccentricity e and α, belonging to the interval (12).

The parameters $\sigma_{1,2}$ are different real numbers only if the inertia parameter α belongs to the interval

$$\frac{8}{9} \leq \alpha \leq 2 \ . \tag{13}$$

This can be readily seen from Fig. 1, where the graphics of functions $\sigma_{1,2}^2(\alpha)$ are shown. Nevertheless, there are ten points in the interval (13), where the inequality (9) is not fulfilled. Actually, at points

$$\alpha_0 = \frac{8}{9},\ 1,\ \frac{4}{3},\ \frac{1}{9}(28-\sqrt{241}),\ \frac{1}{3}(-7+\sqrt{145}),\ \frac{1}{39}(-28+\sqrt{11041}) \tag{14}$$

we have

$$2\sigma_2 = 0,\ 1,\ 2;\ 2\sigma_1 = 3,\ 4,\ 5\ , \tag{15}$$

respectively (see Fig. 2). Besides, at points

$$\alpha_0 = 0.895178,\ 1.16676,\ 1.60055,\ 1.79656 \tag{16}$$

the following equalities are fulfilled

$$\sigma_1 + \sigma_2 = 1,\ 2,\ 3;\ \sigma_1 - \sigma_2 = 1\ . \tag{17}$$

Thus, the domains of instability of the solution (3) in the $\alpha - e$ plane can arise only in the vicinity of the points (14), (16).

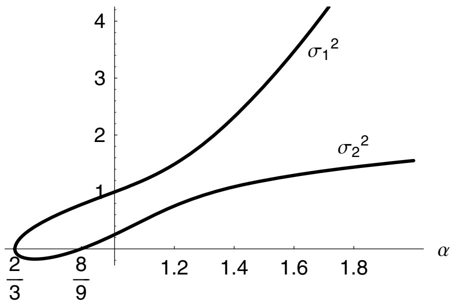


Fig. 1. Parameters $\sigma_{1,2}(\alpha)$ for $\beta = 3/2$

Note that in the cases (15) the corresponding characteristic multipliers $\rho = \exp(2\pi\sigma i)$ for the system (8) are equal to $\rho = 1$ or $\rho = -1$. The important and significant point here is that the cases $\rho = 1$ and $\rho = -1$ are characterized by the existence of periodic solutions of the system (8) with periods 2π and 4π, respectively (see, for example, [13] [14]). Thus, the boundaries between the domains of stability and instability in the $\alpha - e$ plane are some curves $\alpha = \alpha(e)$ which are characterized by the presence of periodic solutions with the periods 2π or 4π and cross the α-axis in the points (14).

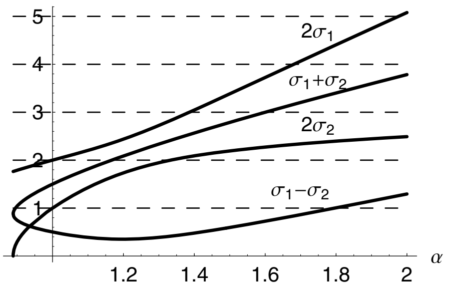


Fig. 2. Dependence of $2\sigma_1,\ 2\sigma_2,\ \sigma_1 \pm \sigma_2$ on α ($\beta = 3/2$)

In order to find the stability boundaries, let us rewrite (8) in the form of two second-order differential equations

$$\begin{aligned}
&(1+e\cos\nu)^2(\ddot{q}_1+\dot{q}_2) - 2e(1+e\cos\nu)\sin\nu(\dot{q}_1+q_2) + \\
&= \left(1-\frac{3}{2}(1-e^2)^{3/2}\alpha + 2e\cos\nu + e^2\cos^2\nu\right)(q_1-\dot{q}_2)\ , \\
&(1+e\cos\nu)^2(\ddot{q}_2-q_1) + 2e(1+e\cos\nu)\sin\nu(q_1-\dot{q}_2) = \\
&= \left(1-\frac{3}{2}(1-e^2)^{3/2}\alpha + 2e\cos\nu + e^2\cos^2\nu\right)(\dot{q}_1+q_2) - \\
&\qquad -3(\alpha-1)(1+e\cos\nu)q_2\ .
\end{aligned} \tag{18}$$

Now we can attempt to seek a solution of the system (18) in the form of Fourier series

$$\begin{aligned}
q_1 &= a_0 + \sum_{k=1}^{\infty}\left(a_k\ \cos\left(\frac{k\nu}{2}\right) + b_k\ \sin\left(\frac{k\nu}{2}\right)\right)\ , \\
q_2 &= p_0 + \sum_{k=1}^{\infty}\left(p_k\ \cos\left(\frac{k\nu}{2}\right) + q_k\ \sin\left(\frac{k\nu}{2}\right)\right)\ .
\end{aligned} \tag{19}$$

Note that (19) correspond to Fourier series for the periodic functions $q_1 = q_1(\nu),\ q_2 = q_2(\nu)$ with the period 4π. But they can also be used to obtain the solution with period 2π by setting to zero the Fourier coefficients corresponding to k being an odd integer. On substituting (19) into (18) and equating coefficients of $\cos(k\nu/2)$ and $\sin(k\nu/2)$ to zero we obtain four infinite sequences of linear algebraic equations for the coefficients of Fourier series (19). The first sequence of equations determines the even coefficients $a_{2k},\ q_{2k}$ and can be written in the form

$$\left(1+\tfrac{1}{2}e^2-\tfrac{3}{2}(1-e^2)^{3/2}\alpha\right)a_0 + ea_2 - eq_2 + \tfrac{1}{4}e^2a_4 - \tfrac{1}{2}e^2q_4 = 0\ ,$$

$$\vdots$$

$$
\begin{gathered}
-\tfrac{1}{4}e^2(k-1)^2 a_{2k-4} + \tfrac{1}{2}e^2(k-1)q_{2k-4} - e(1-k+k^2)a_{2k-2} + \\
+e(2k-1)q_{2k-2} + \tfrac{1}{2}\left(-2-2k^2-e^2(1+k^2)+3(1-e^2)^{3/2}\alpha\right)a_{2k} + \\
+\tfrac{k}{2}\left(4+2e^2-3(1-e^2)^{3/2}\alpha\right)q_{2k} - e(1+k+k^2)a_{2k+2} + \\
+e(2k+1)q_{2k+2} - \tfrac{1}{4}e^2(k+1)^2 a_{2k+4} + \tfrac{1}{2}e^2(k+1)q_{2k+4} = 0 \ , \\
\tfrac{1}{2}e^2(k-1)a_{2k-4} - \tfrac{1}{4}e^2(k-1)^2 q_{2k-4} + e(2k-1)a_{2k-2} + \\
+\tfrac{1}{2}e(-5+2k-2k^2+3\alpha)q_{2k-2} + \tfrac{k}{2}\left(4+2e^2-3(1-e^2)^{3/2}\alpha\right)a_{2k} + \\
+\tfrac{1}{2}\left(-8-2k^2-e^2(1+k^2)+6\alpha+3(1-e^2)^{3/2}\alpha\right)q_{2k} + e(2k+1)a_{2k+2} - \\
-\tfrac{1}{2}e(5+2k+2k^2-3\alpha)q_{2k+2} + \tfrac{1}{2}e^2(k+1)a_{2k+4} - \\
-\tfrac{1}{4}e^2(k+1)^2 q_{2k+4} = 0 \ , \ \ldots
\end{gathered} \tag{20}
$$

The second sequence of equations determines the even coefficients b_{2k}, p_{2k} and is given by

$$
\begin{gathered}
\left(3\alpha-4-\tfrac{1}{2}e^2+\tfrac{3}{2}(1-e^2)^{3/2}\alpha\right)p_0 - eb_2 + \\
+\tfrac{e}{2}(3\alpha-5)p_2 - \tfrac{1}{2}e^2 b_4 - \tfrac{1}{4}e^2 p_4 = 0 \ , \\
\vdots \\
-\tfrac{1}{2}e^2(k-1)b_{2k-4} - \tfrac{1}{4}e^2(k-1)^2 p_{2k-4} - e(2k-1)b_{2k-2} + \\
+\tfrac{1}{2}e(-5+2k-2k^2+3\alpha)p_{2k-2} - \tfrac{k}{2}\left(4+2e^2-3(1-e^2)^{3/2}\alpha\right)b_{2k} + \\
+\tfrac{1}{2}\left(-8-2k^2-e^2(1+k^2)+6\alpha+3(1-e^2)^{3/2}\alpha\right)p_{2k} - e(2k+1)b_{2k+2} - \\
-\tfrac{1}{2}e(5+2k+2k^2-3\alpha)p_{2k+2} - \tfrac{1}{2}e^2(k+1)b_{2k+4} - \tfrac{1}{4}e^2(k+1)^2 p_{2k+4} = 0 \ , \\
-\tfrac{1}{4}e^2(k-1)^2 b_{2k-4} - \tfrac{1}{2}e^2(k-1)p_{2k-4} - e(1-k+k^2)b_{2k-2} - \\
-e(2k-1)p_{2k-2} + \tfrac{1}{2}\left(-2-2k^2-e^2(1+k^2)+3(1-e^2)^{3/2}\alpha\right)b_{2k} - \\
-\tfrac{k}{2}\left(4+2e^2-3(1-e^2)^{3/2}\alpha\right)p_{2k} - e(1+k+k^2)b_{2k+2} - \\
-e(2k+1)p_{2k+2} - \tfrac{1}{4}e^2(k+1)^2 b_{2k+4} - \tfrac{1}{2}e^2(k+1)p_{2k+4} = 0 \ , \ \ldots
\end{gathered} \tag{21}
$$

The third sequence of equations is for the odd coefficients a_{2k-1}, q_{2k-1} and can be written as

$$
\begin{gathered}
\tfrac{1}{2}\left(-\tfrac{5}{2}-\tfrac{3}{2}e-\tfrac{5}{4}e^2+3(1-e^2)^{3/2}\alpha\right)a_1 - \tfrac{1}{4}e\left(7+\tfrac{1}{4}e^2\right)a_3 - \\
-\tfrac{9}{16}e^2 a_5 + \left(1+\tfrac{1}{2}e^2-\tfrac{3}{4}(1-e^2)^{3/2}\alpha\right)q_1 + \left(2e+\tfrac{1}{4}e^2\right)q_3 + \tfrac{3}{4}e^2 q_5 = 0 \ , \\
\left(1+\tfrac{1}{2}e^2-\tfrac{3}{4}(1-e^2)^{3/2}\alpha\right)a_1 + \left(2e-\tfrac{1}{4}e^2\right)a_3 + \tfrac{3}{4}e^2 a_5 + \\
\left(-\tfrac{17}{4}+\tfrac{9}{4}e-\tfrac{5}{8}e^2+3\alpha-\tfrac{3}{2}e\alpha+\tfrac{3}{2}(1-e^2)^{3/2}\alpha\right)q_1 + \\
+\left(-\tfrac{13}{4}e+\tfrac{1}{16}e^2+\tfrac{3}{2}e\alpha\right)q_3 - \tfrac{9}{16}e^2 q_5 = 0 \ , \\
\vdots \\
-\tfrac{1}{16}e\left(e(3-2k)^2 a_{2k-5} + 4(7-8k+4k^2)a_{2k-3} + (12+16k^2)a_{2k+1} + \right. \\
+e(2k+1)^2 a_{2k+3} - 4e(2k-3)q_{2k-5} - 32(k-1)q_{2k-3} - 32kq_{2k+1} - \\
\left. -\ 4e(2k+1)q_{2k+3}\right) + \tfrac{1}{4}(2k-1)(4+2e^2-3(1-e^2)^{3/2}\alpha)q_{2k-1} + \\
+\tfrac{1}{8}\left(-10+8k-8k^2+e^2(-5+4k-4k^2)+12(1-e^2)^{3/2}\alpha\right)a_{2k-1} = 0 \ ,
\end{gathered}
$$

$$-\tfrac{1}{16}e\left(4e(3-2k)a_{2k-5}+e(3-2k)^2q_{2k-5}-32(k-1)a_{2k-3}-32ka_{2k+1}-\right.$$
$$-4e(2k+1)a_{2k+3}+4(13-8k+4k^2-6\alpha)q_{2k-3}+(36+16k^2-24\alpha)q_{2k+1}+$$
$$\left.+\,e(2k+1)^2q_{2k+3}\right)+\tfrac{1}{4}(2k-1)(4+2e^2-3(1-e^2)^{3/2}\alpha)a_{2k-1}+$$
$$+\tfrac{1}{8}\left(-34+8k-8k^2+e^2(-5+4k-4k^2)+24\alpha+\right.$$
$$\left.+\,12(1-e^2)^{3/2}\alpha\right)q_{2k-1}=0\ ,\ \dots \tag{22}$$

At last, the fourth sequence of equations is for the odd coefficients b_{2k-1}, p_{2k-1} and is given by

$$\left(-\tfrac{5}{4}+\tfrac{3}{4}e-\tfrac{5}{8}e^2+\tfrac{3}{2}(1-e^2)^{3/2}\alpha\right)b_1-\tfrac{1}{4}e\left(7-\tfrac{1}{4}e^2\right)b_3-$$
$$-\tfrac{9}{16}e^2b_5+\left(-1-\tfrac{1}{2}e^2+\tfrac{3}{4}(1-e^2)^{3/2}\alpha\right)p_1+\left(-2e+\tfrac{1}{4}e^2\right)p_3-\tfrac{3}{4}e^2p_5=0\ ,$$
$$\left(-1-\tfrac{1}{2}e^2+\tfrac{3}{4}(1-e^2)^{3/2}\alpha\right)b_1-\left(2e+\tfrac{1}{4}e^2\right)b_3-\tfrac{3}{4}e^2b_5+$$
$$\left(-\tfrac{17}{4}-\tfrac{9}{4}e-\tfrac{5}{8}e^2+3\alpha+\tfrac{3}{2}e\alpha+\tfrac{3}{2}(1-e^2)^{3/2}\alpha\right)p_1-$$
$$-\left(\tfrac{13}{4}e+\tfrac{1}{16}e^2-\tfrac{3}{2}e\alpha\right)p_3-\tfrac{9}{16}e^2p_5=0\ ,$$
$$\vdots$$
$$-\tfrac{1}{16}e\left(4e(3-2k)b_{2k-5}+e(3-2k)^2p_{2k-5}+32(k-1)b_{2k-3}+32kb_{2k+1}+\right.$$
$$+4e(2k+1)b_{2k+3}+4(13-8k+4k^2-6\alpha)p_{2k-3}+(36+16k^2-24\alpha)p_{2k+1}+$$
$$\left.+\,e(2k+1)^2p_{2k+3}\right)-\tfrac{1}{4}(2k-1)(4+2e^2-3(1-e^2)^{3/2}\alpha)b_{2k-1}+$$
$$+\tfrac{1}{8}\left(-34+8k-8k^2+e^2(-5+4k-4k^2)+24\alpha+12(1-e^2)^{3/2}\alpha\right)p_{2k-1}=0\ ,$$
$$-\tfrac{1}{16}e\left(e(3-2k)^2b_{2k-5}+4(7-8k+4k^2)b_{2k-3}+(12+16k^2)b_{2k+1}+\right.$$
$$+e(2k+1)^2b_{2k+3}+4e(2k-3)p_{2k-5}+32(k-1)p_{2k-3}+32kp_{2k+1}+$$
$$\left.+\,4e(2k+1)p_{2k+3}\right)-\tfrac{1}{4}(2k-1)(4+2e^2-3(1-e^2)^{3/2}\alpha)p_{2k-1}+$$
$$+\tfrac{1}{8}\left(-10+8k-8k^2+e^2(-5+4k-4k^2)+\right.$$
$$\left.+\,12(1-e^2)^{3/2}\alpha\right)b_{2k-1}=0\ ,\ \dots \tag{23}$$

Each of the sequences (20)-(23) is just an infinite system of linear homogeneous equations. It is known that such a system has a non-trivial solution only if determinant of the corresponding matrix of the system is equal to zero. This condition gives the equation determining the stability boundaries in the $\alpha - e$ plane. Obviously, these boundaries must reduce to (14) when $e \to 0$.

Of course, it is impossible to calculate a determinant of the infinite matrix. So, in order to find the stability boundaries $\alpha = \alpha(e)$ we should truncate the infinite sequences of equations (20)-(23) after the kth term, where k is a suitably large number. The corresponding determinant for the system determining the coefficients a_{2k}, q_{2k} , for instance, can be written as $(k = 1)$

$$D_k = \begin{vmatrix} -1-\frac{1}{2}e^2+\frac{3}{2}\mu & -e & e \\ -2e & -2-e^2+\frac{3}{2}\mu & 2+e^2-\frac{3}{2}\mu \\ 2e & 2+e^2-\frac{3}{2}\mu & -5-e^2+3\alpha+\frac{3}{2}\mu \end{vmatrix}\ , \tag{24}$$

where $\mu = (1-e^2)^{3/2}\alpha$.

Equating determinant of the form (24) to zero, we obtain an algebraic equation giving an approximation for the stability boundary $\alpha = \alpha(e)$. An exact expression for the boundary is obtained when $k \to \infty$. This approach giving the equation for the stability boundary is known as the infinite determinant method [9].

It can be readily seen from (24) that in the case of $e = 0$ determinants of systems (20)-(23) will be equal to zero when α takes values from (14). It means that the stability boundaries cross the $e = 0$ axis in the $\alpha - e$ plane at the points (14). For sufficiently small e we can represent the corresponding curves $\alpha = \alpha(e)$ in the vicinity of the points (14) as power series

$$\alpha = \alpha_0 + \alpha_1 e + \alpha_2 e^2 + \dots \ , \tag{25}$$

where α_0 is equal to one of the values from (14).

Now we should substitute (25) into the expressions for D_k and expand them in powers of e. Afterwards, equating coefficients of e^k $(k = 1, 2, 3, \dots)$ to zero, we obtain a system of algebraic equations determining the coefficients α_k in the expansion (25). As a result we have found the following curves

$$\alpha = \frac{8}{9} - \frac{4}{27}e^2 + \frac{1}{2349}e^4 - \frac{1051835}{15122862}e^6 \ , \tag{26}$$

$$\alpha = 1 \ , \tag{27}$$

$$\alpha = \frac{4}{3} - e^2 + \frac{1215}{112}e^4 - \frac{537502421}{3951360}e^6 \ , \tag{28}$$

$$\alpha = \frac{4}{3} - \frac{5}{3}e^2 + \frac{5773}{336}e^4 - \frac{334463897}{1317120}e^6 \ , \tag{29}$$

$$\alpha = 1.68053 + 3.26683e^2 - 14.7417e^4 + 97.2646e^6 \ , \tag{30}$$

$$\alpha = 1.68053 + 3.26683e^2 - 22.2214e^4 + 217.149e^6 \ , \tag{31}$$

$$\alpha = 1.3862 + 0.615632e^2 \pm 1.24701e^3 + 1.26813e^4 \pm 1.12375e^5 - 1.32848e^6 \ , \tag{32}$$

$$\alpha = 1.97631 - 2.79566e^2 + 15.4468e^4 \pm 0.678768e^5 - 120.102e^6 \ . \tag{33}$$

It should be emphasized that all coefficients in the expansions (26)-(33) are calculated exactly. But in (30)-(33) they are quite cumbersome and so their approximations are written. We have used here the determinants D_8. If we increase the order of the matrix whose determinant we have calculated in order to find the coefficients in the expansion (25) then we'll find the higher order corrections in (26)-(33). The coefficients α_k that we have already found in (26)-(33) will be the same.

Let us remember now that at points (16) the inequality (9) is also not satisfied. But in these cases there are no periodic solutions of system (18) having the form (19). Nevertheless, according to the Floquet–Liapunov theory (see, for example, [15]), there must exist a solution of the form

$$\begin{aligned} q_1 &= \exp(\lambda\nu)\left(a_0 + \sum_{k=1}^{\infty}\left(a_k\ \cos(k\nu) + b_k\ \sin(k\nu)\right)\right) \ , \\ q_2 &= \exp(\lambda\nu)\left(p_0 + \sum_{k=1}^{\infty}\left(p_k\ \cos(k\nu) + q_k\ \sin(k\nu)\right)\right) \ . \end{aligned} \tag{34}$$

Characteristic exponents λ depend on the parameter ε and in the neighborhood of the points (16) can be represented in the form

$$\begin{aligned} \lambda_{1,2} &= \pm i\ (\sigma_1 + \sigma_{11}e + \sigma_{12}e^2 + \ldots) \ , \\ \lambda_{3,4} &= \pm i\ (\sigma_2 + \sigma_{21}e + \sigma_{22}e^2 + \ldots) \ . \end{aligned} \tag{35}$$

Now the boundaries between the domains of stability and instability in the $\alpha - e$ plane are determined from the conditions that coefficients in expansions (35) are real numbers and the following equalities are satisfied

$$\lambda_1 + \lambda_3 = i,\ 2i,\ 3i;\quad \lambda_1 - \lambda_3 = i \ , \tag{36}$$

It can be readily seen that these equalities reduce to (17) for $e \to 0$. Again for sufficiently small e we can represent the stability boundaries $\alpha = \alpha(e)$ in the neighborhood of the points (16) as power series (25). In order to find the corresponding curves we should substitute (25) and (35) into (18) and taking into account (36) repeat all the calculations we have done above. It should be noted that the calculations in these cases are much more bulky and we can do them only with the modern computer algebra system such as *Mathematica*, for example. As a result we have found the following curves

$$\begin{aligned} \alpha = 0.895178 \pm 0.068858e + 0.122736e^2 \pm 0.0683111e^3 - \\ -0.559085e^4 \pm 0.748201e^5 - 0.510311e^6 \ , \end{aligned} \tag{37}$$

$$\alpha = 1.16676 - 0.528825e^2 + 12.5091e^4 \ , \tag{38}$$

$$\alpha = 1.16676 + 0.370905e^2 - 3.48835e^4 \ , \tag{39}$$

$$\alpha = 1.60055 - 0.103666e^2 - 65.6796e^4 \ , \tag{40}$$

$$\alpha = 1.60055 + 2.51269e^2 - 21.5334e^4 \ , \tag{41}$$

$$\alpha = 1.79656 \pm 0.377688e - 2.47355e^2 \pm 64.3954e^3 + 991.319e^4 \ . \tag{42}$$

Remind that the values of α_0 in (37)-(42) are determined as solutions of (17). Using approximate values of α_0 and σ_1, σ_2, we have found approximate values of the corresponding coefficients α_k in (23) as well. But using the system *Mathematica* we can calculate them with arbitrary precision.

4 Conclusion

In the present paper we study the stability of cylindrical precession of the dynamically symmetric satellite in the Newtonian gravitational field. In the case of the resonance 3 : 2 (Mercury type resonance) we have shown that cylindrical precession of the satellite may be stable only if the inertia parameter α belongs to the interval (13). It turned out that there are ten values of the parameter α in the interval (13) where conditions of the second order resonance

$$\sigma_j \pm \sigma_k = N \ (j,k = 1,2;\ N = 0,\ \pm 1,\ \pm 2,\ \ldots) \qquad (43)$$

are fulfilled and the domains of instability can arise in the vicinity of these points in the $\alpha - e$ plane. Using the infinite determinant method, we have developed the algorithm of analytical calculation of the boundaries of the domains of instability and found these boundaries analytically in the form of power series in a small parameter e that is just the eccentricity of the satellite orbit. The obtained results show that a bandwidth of the domains of instability is $O(e^N)$ and decreases very fast if the number N is growing up. These results are in agreement with the general theory of parametric resonance (see, for example, [16]). All the calculations and visualization of the obtained results have been done with the computer algebra system *Mathematica*.

References

1. Gerdt, V.P., Tarasov, O.V., Shirkov, D.V.: Analytic calculations on digital computers for applications in physics and mathematics. Usp. Fiz. Nauk **130**, No. 1 (1980) 113–147 (in Russian)
2. Beletskii, V.V.: The motion of an artificial satellite about its center of mass in a gravitational field. Moscow Univ. Press (1975) (in Russian)
3. Markeev, A.P., Chekhovskaya, T.N.: On the stability of cylindrical precession of a satellite on the elliptic orbit. Prikl. Math. Mech. **40** (1976) 1040–1047 (in Russian)
4. Churkina, T.E.: On stability of a satellite motion in the elliptic orbit in the case of cylindrical precession. Mathematical Modelling **16**, No. 7 (2004) 3–5 (in Russian)
5. Markeev, A.P.: The stability of hamiltonian systems. In: Nonlinear mechanics, V.M.Matrosov, V.V.Rumyantsev, A.V.Karapetyan (Eds.). Fizmatlit, Moscow (2001) 114–130 (in Russian)
6. Markeev, A.P.: The libration points in celestial mechanics and cosmic dynamics. Nauka, Moscow (1978) (in Russian)
7. Wolfram, S.: The Mathematica Book. 4th edn. Wolfram Media/Cambridge University Press (1999)
8. Yakubovich, V.A., Starzhinskii, V.M.: Linear Differential Equations with Periodic Coefficients. John Wiley, New York (1975)
9. Lindh, K.G., Likins, P.W.: Infinite determinant methods for stability analysis of periodic-coefficient differential equations. AIAA J. **8** (1970) 680–686
10. Prokopenya, A.N.: Studying stability of the equilibrium solutions in the restricted many-body problems. In: Challenging the Boundaries of Symbolic Computation, Proc. 5th Int. Mathematica Symposium (London, Great Britain, 2003), P.Mitic, Ph.Ramsden, J.Carne (Eds.). Imperial College Press, London (2003) 105–112

11. Prokopenya, A.N.: Determination of the stability boundaries for the hamiltonian systems with periodic coefficients. Math. Modelling and Analysis **10**, No. 2 (2005) 191–204
12. Liapunov, A.M.: General problem about the stability of motion. Gostekhizdat, Moscow (1950) (in Russian)
13. Merkin, D.R.: Introduction to the Theory of Stability. Springer-Verlag, Berlin, New York (1997)
14. Grimshaw, R.: Nonlinear Ordinary Differential Equations. CRC Press (2000)
15. Cesari, L.: Asymptotic behaviour and stability problems in ordinary differential equations. 2nd edn. Academic Press, New York (1964)
16. Landau, L.D., Lifshits, E.M.: Theoretical Physics, Vol. 1. Mechanics. 4th edn. Nauka, Moscow (1988) (in Russian)

Generation of Orthogonal Grids on Curvilinear Trimmed Regions in Constant Time

Dmytro Chibisov, Victor Ganzha, Ernst W. Mayr[1], and Evgenii V. Vorozhtsov[2]

[1] Institute of Informatics, Technical University of Munich,
Garching 85748, Boltzmannstr. 3, Germany
{chibisov, ganzha, mayr}@in.tum.de
[2] Institute of Theoretical and Applied Mechanics,
Russian Academy of Sciences, Novosibirsk 630090, Russia
vorozh@itam.nsc.ru

Abstract. We propose a new algorithm for the generation of orthogonal grids on regions bounded by arbitrary number of polynomial inequalities. Instead of calculation of the grid nodes positions for a particular region, we perform all calculations for general polynomials given with indeterminate coefficients. The first advantage of this approach is that the calculations can be performed only once and then used to generate grids on arbitrary regions and of arbitrary mesh size with constant computational costs. The second advantage of our algorithm is the avoidance of singularities, which occur while using the existing algebraic grid generation methods and lead to the intersection of grid lines. All symbolic calculation can be performed with general purpose Computer Algebra Systems, and expressions obtained in this way can be translated in Java/C++ code.

1 Motivation and Introduction

Advanced computer technologies and parallel architectures allow one to solve time dependent problems with 10^9 and more unknowns on rectangular regions in realistic time using hierarchical and adaptive approaches [3,13]. In order to handle problems of such order of computational complexity on arbitrary regions and, in particular, with moving boundaries, we are interested to have efficient grid generation techniques, which would support hierarchical approach to computing and provide the possibility of adaptive grid refinement without additional computational costs.

Grid generation techniques may be subdivided into two big classes:
(a) techniques based on the numerical solution of certain PDEs;
(b) algebraic techniques [8,12].

Among the techniques belonging to class (a), the grid generation techniques based on the solution of elliptic partial differential equations (PDEs) have gained the most widespread acceptance. We should, however, mention the following two shortcomings of these techniques.

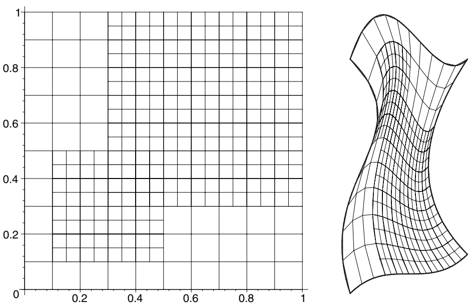


Fig. 1. Adaptivity of transfinite interpolation

- They require an iterative solution of finite difference equations approximating the elliptic PDEs. This may require a CPU time comparable with the CPU time needed for the solution of the fluid dynamics problem on the generated curvilinear grid.
- The intersections of grid lines of the same family may occur near the sharp corners of the boundary or the boundary intervals with high curvature. This is inadmissible from the viewpoint of the subsequent solution of fluid dynamics equations on such a grid.

The numerical grid generation techniques based on the numerical solution of hyperbolic [8,12,11,10] and parabolic [8,9] equations indeed belong to the *marching* techniques of grid generation. Therefore, these techniques are very fast compared to methods that require solving second-order elliptic systems. There are, however, limitations inherent in hyperbolic and parabolic grid generation. These methods apply only to unbounded domains, i.e., the outer boundary cannot be specified in advance.

In contrast to PDE's based grid generation methods, the algebraic grid generation (AGG) makes possible the hierarchical and adaptive methods. AGG methods can be described as polynomial mappings from unit square to the region of interest built from boundary polynomials. Therefore, the AGG techniques have two main advantages:

– since the analytic mapping for a particular region is derived only once, computational costs are independent of the number of grid nodes
– for the same reason, hierarchical adaptive refinement is supported in a natural way (Fig. 1)

The algebraic grid generation techniques are much faster than the techniques based on the numerical solution of elliptic PDEs. The most well known representative techniques in this class are the multi-surface method [6,7] and the method of transfinite interpolation [8,12]. The shortcoming of the multi-surface method [6,7] is that it involves many adjustable parameters controlling the distribution of grid nodes on the spatial region boundaries. It is impossible in practice to guess the values of these parameters in such a way that the generated curvilinear grid would be orthogonal or nearly orthogonal.

A shortcoming of the grid generation by transfinite interpolation is that this technique does not incorporate any check-up of the grid orthogonality.

One of further disadvantages of AGG are possible singularities, which lead to overspill phenomenon shown in Fig. 2.

In the present paper, we propose a new algorithm for numerical grid generation, which belongs to the class of marching techniques and which incorporates explicitly the orthogonality property of the curvilinear grid. In addition, our algorithm, as will be shown by the examples, is applicable both to bounded and unbounded domains. Since our algorithm belongs to the class of the AGG techniques, it preserves all the above mentioned advantages of AGG.

We present namely an algorithm that computes an orthogonal grid on the region, whose boundaries are given by N parametric curves $[x^{(j)}(t), y^{(j)}(t)], t \in [0, 1]$ of degree $deg \leq M$ with symbolic coefficients b_i^j:

$$\begin{array}{ll} x_1(t) = \sum_{i=1}^{M} a_i^{(1)} t^i & y_1(t) = \sum_{i=1}^{M} b_i^{(1)} t^i \\ \ldots & \\ x_N(t) = \sum_{i=1}^{M} a_i^{(N)} t^i & y_N(t) = \sum_{i=1}^{M} b_i^{(N)} t^i \end{array} \tag{1}$$

We desrcribe in Section 2 how starting from these equations, the grid lines can be calculated using the notion of *envelope*. The fundamental problem in this approach is the efficient localization of singularities, which result in self-intersection of grid lines. In [5], the Groebner Bases were used to find the singularities of envelopes. However, in the presence of symbolic coefficients in (1) this approach becomes computationally too expensive. For example, we have started the calculation of Groebner Bases for the localization of singularities of envelopes according to [5], but with symbolic coefficients, in Maple and interrupted it after 72 hours of computation without any result. In Section 3 we will show how this task can be solved in realistic time using the resultants. The well-known problem is that the resultants tend to become very large and may contain many thousands of terms even in the case of 3 equations in 2 variables of degree 2. Therefore, we describe a way to represent the resultants arising in grid generation problem in a compact way.

2 Our Approach

In the present section, we show how the orthogonal grid can be generated using algebraic methods. Let the region Ω (Figs. 1, 2) be bounded by the roots of polynomials $f_i(x, y)$. The so-called Tarski formula describing the set of points, which belong to this region can be written as follows:

$$\Omega(x, y) \equiv \bigwedge_i f_i(x, y) \geq 0$$

We propose to calculate the lines of the curvilinear grid in the following way. We contact a circle $C(x, y) = x^2 + y^2 - r^2 \geq 0$ with Ω and move C along the

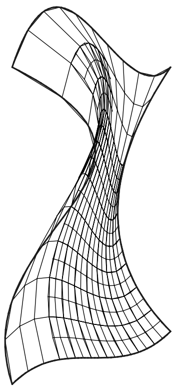


Fig. 2. Overspill phenomenon

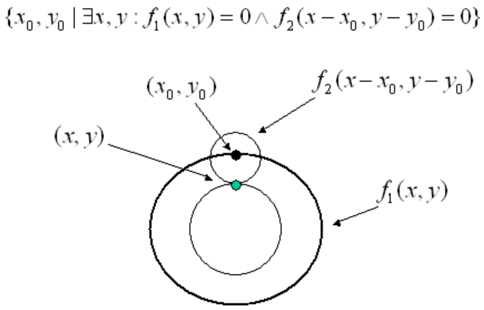


Fig. 3. Calculating of envelopes by quantifier elimination

boundary of Ω keeping them in contact. The motion of a circle can be produced by shifting it by x_0, y_0 units:

$$C(x - x_0, y - y_0) \geq 0.$$

The circle moving along some boundary curve $f_i(x, y) = 0$ describes a curve $g_i(x, y) = 0$ called *envelope* (Fig. 3). More precisely, the envelope in our case is a curve, whose tangent at each point coincides with the tangent of a moving circle at each time of its motion. Connecting contact points of C with f_i and g_i with C for some x_0, y_0 produces a line segment, which is orthogonal to both curves. Proceeding in this way the orthogonal grid lines can be calculated.

The contact of C and f_i can be expressed in terms of common roots of bounding polynomials. The g_i corresponds also to such shifts x_0, y_0 of C, where polynomial f_i and C have common roots. This can be formalized with Tarski sentences involving only equations as follows:

$$\{(x_0, y_0) | \exists x, y : f(x, y) = 0 \wedge C(x - x_0, y - y_0) = 0.\} \tag{2}$$

Eliminating $\exists$-quantifiers with existing methods described below produces the point set, which corresponds to the envelope g (Fig. 3). After $g(x, y)$ is calculated, the points distributed along them should be connected with those of $f(x, y)$ in such a way the connecting line segment is normal to both curves.

2.1 Grid Generation by Elimination of Variables

In this section we shall describe how the well known approach to the elimination of variables from the following first-order formulas (so-called existential first-order theory over the reals) with the aid of resultants can be used to generate orthogonal grids:

$$\exists x_1, ..., x_M : \bigwedge_i \bigvee_j f_{i,j}(x_1, ..., x_M, ..., x_N) = 0.$$

For example, the curve described by a circle rolling along parametric curves (1) can be described with the following formula:

$$\exists t : \bigvee_{i=1}^{N} C(x^{(i)}(t) - x_c, y^{(i)}(t) - y_c, r) = 0, \qquad (3)$$

where C is the circle equation with indeterminate radius r and center position x_c, y_c. Given a polynomial $f(x)$ of degree n with roots α_i and a polynomial $g(x)$ of degree m with roots β_j, the resultant is defined by

$$\rho(f, g) = \prod_{i,j} (\alpha_i - \beta_j).$$

$\rho(f, g)$ vanishes iff $\exists a : f(a) = 0 \wedge g(a) = 0$. The resultant can be computed as the determinant of the so-called Sylvester Matrix [5]. In the multivariate case, the computation of resultant can be reduced to the univariate one by considering the polynomials $f, g \in \mathbb{K}[x_1, ..., x_N]$ as univariate polynomials in $\mathbb{K}[x_1]$ with unknown coefficients in $\mathbb{K}[x_2, ..., x_N]$ (denoted as $\mathbb{K}(x_2, ..., x_N)[x_1]$). In the following we call the resultant of $f, g \in \mathbb{K}(x_1, ..., x_{i-1}, x_{i+1}, ..., x_N)[x_i]$ as $res_{x_i}(f, g)$.

In order to eliminate t from (2) and find the envelope h of the circle and (1) we may calculate

$$h(x_c, y_c, r) = res_t \left(C(x(t) - x_c, y(t) - y_c, r), \frac{\partial}{\partial t} C(x(t) - x_c, y(t) - y_c, r) \right) \qquad (4)$$

In this way the first family of grid lines, namely parallel to the boundary, can be calculated symbolically. Note that the resultant for a curve (1) with symbolic coefficients has already 2599 terms even if the degree is equal to 2. As will be described in Section 3, in order to find a self-intersection of grid lines further resultants of such large expressions will be needed. Therefore, we will present in Section 3.1 a much more efficient way to localize the self-intersections of such curves without computing resultants completely based on polynomial remainder sequences.

In order to generate the second family of grid lines, which are perpendicular to the first family, we discretize each of the curves (1) with step size $\Delta t = \frac{1}{N}$ and obtain a number of points $p_j = (x_j, y_j), j = 1..N$. This can be done with symbolic N by substitution of $t = \frac{j}{N}, j = 1, \ldots, N$ in (1). We place the circle center in each p_j and compute the intersection of $C(x-x_j, y-y_j)$ with $h(x, y, r)$. Let us denote the common roots of the both polynomials as $\mathbf{V}(C(x-x_j, y-y_j), h(x, y, r))$. Thus, the second family of grid lines $v(j, r)$ perpendicular to $h(x, y, r) = 0$ can be obtained by computing

$$v(j, r) = \mathbf{V}(C(x-x_j, y-y_j), h(x, y, r)). \tag{5}$$

Since $h(x, y, r)$ is a large symbolic expression, as mentioned previously, computing (5) in a direct way by elimination of variables using resultants becomes a very expensive task.
Therefore, we use the following simple result, which gives the intersection of a circle with middle point $(x_j, y_j) \in [x(t), y(t)]$ and radius r and $h(x, y, r)$ given by (4):

Proposition 1. *Let $h(x_c, y_c, r)$ be envelope of a family of circles $C(x-x(t), y-y(t), r)$ with radius r given by (4). Then for any $t_j \in \mathbb{R}$ the following is satisfied:*

$$\mathbf{V}(C(x-x(t_j), y-y(t_j), r), h(x, y, r)) =$$
$$\left(x(t_j) \pm \frac{r\frac{dy}{dt}|_{t_j}}{\sqrt{\frac{dy}{dt}|_{t_j} + \frac{dx}{dt}|_{t_j}}}, \quad y(t_j) \mp \frac{r\frac{dx}{dt}|_{t_j}}{\sqrt{\frac{dx}{dt}|_{t_j} + \frac{dx}{dt}|_{t_j}}} \right).$$

Proof. According to (4) $\mathbf{V}(C(x-x(t_j), y-y(t_j)), h(x, y, r)) = \mathbf{V}(C(x-x(t), y-y(t), r), \frac{\partial}{\partial t}C(x-x(t), y-y(t), r))$ for some t. Note that

$$\frac{\partial}{\partial t}C(x-x(t), y-y(t), r) = \frac{\partial C}{\partial x}\frac{dx}{dt} + \frac{\partial C}{\partial y}\frac{dy}{dt} = -2(x-x(t))\frac{dx}{dt} - 2(y-y(t))\frac{dy}{dt}.$$

This means that all solutions of $\mathbf{V}(C(x-x(t_j), y-y(t_j), r), h(x, y, r))$ lie on a line $x\frac{dx}{dt}|_{t_j} + y\frac{dy}{dt}|_{t_j} - x(t_j)\frac{dx}{dt}|_{t_j} - y(t_j)\frac{dy}{dt}|_{t_j} = 0$ independently of r. Thus, we are interested to find the intersections of circle $C(x-x_j, y-y_j, r)$ and this line going through the middle point of C. Using a bit of elementary mathematics we obtain the statement of this proposition. $\diamondsuit$

Now we are able to find the points (x_h, y_h) on envelope $h(x, y, r)$, which correspond to the particular position (x_b, y_b) of a circle on the boundary of the region. For example, when the bounding curve (1) is of degree 3 with unknown coefficients $a_1, ..., a_4, b_1, ..., b_4$ we obtain using Proposition 1:

$$\begin{aligned}
x_b &= a_1 + a_2 t_j + a_3 {t_j}^2 + a_4 {t_j}^3 \\
y_b &= b_1 + b_2 t_j + b_3 {t_j}^2 + b_4 {t_j}^3 \\
x_h &= a_1 + a_2 t_j + a_3 {t_j}^2 + a_4 {t_j}^3 + \frac{r\left(b_2 + 2\,b_3 t_j + 3\,b_4 t_j^2\right)}{\sqrt{{b_2}^2 + 4\,b_2 b_3 t_j + 4\,{b_3}^2 {t_j}^2 + {a_2}^2 + 4\,a_2 a_3 t_j + 4\,{a_3}^2 {t_j}^2}} \\
y_h &= b_1 + b_2 t_j + b_3 {t_j}^2 + b_4 {t_j}^3 - \frac{r\left(a_2 + 2\,a_3 t_j + 3\,a_4 t_j^2\right)}{\sqrt{{b_2}^2 + 4\,b_2 b_3 t_j + 4\,{b_3}^2 {t_j}^2 + {a_2}^2 + 4\,a_2 a_3 t_j + 4\,{a_3}^2 {t_j}^2}}.
\end{aligned} \tag{6}$$

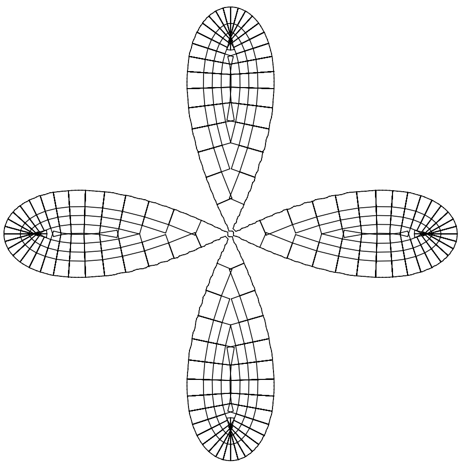


Fig. 4. Generated Grid

The line segment connecting points (x_b, y_b) on the boundary and points $(x_h(r), y_h(r))$ on $h(x, y, r)$ which is orthogonal to boundary curve as well to $h(x, y, r)$ for all r can now be obtained using (6). For example, the grid shown in Fig. 4 has been generated in this way.

So far we have considered successive generation of grid cells for an individual boundary curve by computing two families of grid lines: perpendicular and parallel to this curve. Since the given region is bounded by several trimmed curves, it is convenient to provide a method guaranteeing that the edges of grid cells generated for both curves do not intersect or even coincide in their nodes. As can easily be seen, the initial distribution of points (x_b, y_b) on the boundary is critical for this purpose. Therefore, we propose the following approach that guarantees

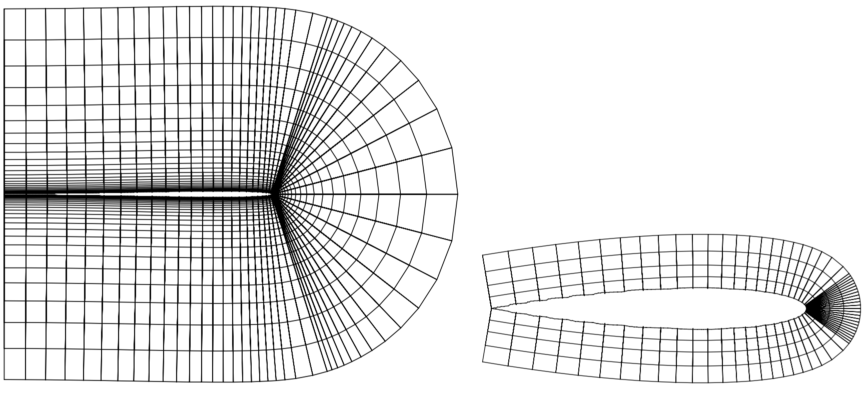


Fig. 5. Generated grid for `NACA-00t′` profiles: complete C-grid (on the left), partial view (on the right)

that mesh cells generated along two given intersecting curves $[x_1(t), y_1(t)]$ and $[x_2(t), y_2(t)]$ do not intersect and, so far the angles are suitable, coincide in their nodes.

We start from any initial distribution of boundary points $p_j^{(1)} \in [x_1(t), y_1(t)]$ and compute such points $p_k^{(2)} \in [x_2(t), y_2(t)]$ that the generated cells coincide in their nodes using (6).

Using this approach we have computed the so-called C-grids about the aerofoil profiles of the popular family `NACA-00t′`.

In the next section we shall show how possible intersections of obtained line segments, which lead to the self-intersection of grid lines, can be detected and avoided.

3 Handling Self-intersections

As shown in Fig. 2, the fundamental problem of the AGG methods is the self-intersection of grid lines. In this section we propose the way to overcome this difficulty.

Let $h(x_1, ..., x_n, r) : \mathbf{R}^{n+1} \to \mathbf{R}$ be a family of grid lines calculated in the previous section. Consider the Jacobian at point $\mathbf{p} = (x_1, ..., x_n) \in \mathbf{R}^n$:

$$\mathbf{J}|_{\mathbf{p}} = \left(\frac{\partial h(x_1, ..., x_n, r)}{\partial x_1}|_{\mathbf{p}}, ..., \frac{\partial h(x_1, ..., x_n, r)}{\partial x_n}|_{\mathbf{p}} \right).$$

Recall that the critical points $\mathbf{p} = (x_1, ..., x_n, r) \in \mathbf{R}^{n+1}$ of h, which include selfintersection points, are characterized by $\mathbf{J}|_p = 0$. The set of critical points of $h(x_1, ..., x_n, r) = 0$, denoted $\mathcal{C}_r$, can also be described as common roots of h and its derivatives

$$\mathcal{C}_r = \mathbf{V}(h(x_1, ..., x_n, r), \frac{\partial h(x_1, ..., x_n, r)}{\partial x_1}, ..., \frac{\partial h(x_1, ..., x_n, r)}{\partial x_n}) \tag{7}$$

Unfortunately, the solution of (7), where $h(x_1, ..., x_n, r)$ of degree at least 6 has symbolic coefficients and many thousands of terms, as in our case, is a computationally expensive task. In order to solve this problem in an efficient way we propose the following method based on polynomial division. First of all observe that the selfintersections of the envelope h are exactly the points, where $C(x_1 - x_1(t), ..., x_n - x_n(t), r) = h(x_1, ..., x_n, r)$ for more than one t. Thus, in order to find selfintersection it is sufficient to examine for which values of $x_1, ..., x_n$ the following system of equations has more than one solution with respect to t ([2]):

$$\begin{aligned} C(x_1 - x_1(t), ..., x_n - x_n(t), r) = 0 \\ \frac{\partial}{\partial t} C(x_1 - x_1(t), ..., x_n - x_n(t), r) = 0 \end{aligned}$$

For this purpose we shall use Polynomial Remainder Sequences (PRS) that generalizes the notion of resultants. If $f(x)$ and $g(x)$ are polynomials, let the

PRS of f and g be denoted by $r_0, r_1, \ldots, r_M$, where $r_0(x) = f(x), r_1(x) = g(x)$ and the intermediate remainders are computed via

$$r_{i+1} = q_i r_i - r_{i-1},$$

where q_i is the quotient of the polynomial division of r_{i-1} by r_i. ([4], [1]). It is obvious that PRS has finitely many non-zero elements.

Let $SC(f, g, v)$ denote the number of sign changes in the sequence $r_0(v), \ldots,$ $r_M(v)$. Let $SC(f, g, a, b)$ be the quantity $SC(f, g, b) - SC(f, g, a)$. The classical Sturm theorem states that $SC(f, \frac{\partial f}{\partial x}, a, b)$ is equal to the number of real roots of $f(x) = 0$ in interval $[a, b]$. In this way the selfintersection points of the envelope of (1) can be characterized by $SC(C, \frac{\partial}{\partial t} C, 0, 1) \geq 2$. Computing this can be done as described, for example, in [1]. In this way, polynomial remainders can be used in order to localize and prevent self-intersections of grid lines.

4 Conclusion

We have presented the grid generation algorithm that allows us to generate an orthogonal grid on curvilinear trimmed regions bounded by polynomial equations given with symbolic coefficients. In contrast to existing approaches to grid generation, our method supports at the same time exact boundary representation as well as the possibility of adaptive computation. A hierarchical refinement of the grid does not change already generated grid. Furthermore, because of the use of symbolic coefficients in boundary polynomials, all computational work is performed only once, and the generated symbolic expressions are used to obtain the orthogonal grid on any particular region in constant time by simple substitution of coefficients of bounding polynomials. One of the advantages of our method is the avoiding of singularities resulting in self-intersection of grid lines. In order to localize and remove the self-intersection of curves given with indeterminate coefficients in realistic time and space we have presented a number of optimized symbolical techniques based on polynomial division.

The requirement of polynomial boundary curves is a limitation for grid generation by the above proposed method. It is, however, to be noted that there are many applied problems in which the boundaries are described by polynomial functions.

The future work will be concerned with further optimization of used symbolic algorithms from computational point of view in order to allow a very efficient grid generation in higher dimensions.

It is also to be noted that the above presented method can be extended for the case of non-polynomial boundary curves. In this case one has, however, to check after the determination of each new grid line segment whether it has the intersections with all already available grid line segments, so that this procedure can take too much computer time at the numerical generation of fine orthogonal grids. This suggests a direction for future work: a search for the way to circumvent this laborious procedure.

References

1. Basu, S., Pollack, R., Roy M.-F.: Algorithms in Real Algebraic Geometry, Springer-Verlag, 2003
2. Bruce, J.W., Giblin, P.J.: Curves and Singularities, Cambridge University Press, 1984
3. Bungartz, H.-J.: Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung. Dissertation, Institut für Informatik, Technische Universitait München, 1992
4. Canny, J.: Improved algorithms for sign-determination and existential quantifier elimination. Computer Journal **36** (1993) 409-418
5. Cox, D.A., Little, J.B., O'Shea, D.: Ideals, Varieties, and Algorithms, Springer-Verlag, 1996
6. Eiseman, P.R.: A multi-surface method of coordinate generation. J. Comp. Phys. **33** (1979) 118–150
7. Fletcher, C.A.J.: Computational Techniques for Fluid Dynamics, 3rd Edition. Springer-Verlag, Berlin, Heidelberg, New York, 1996
8. Knupp, P., Steinberg, S.: Fundamentals of Grid Generation. CRC Press, Boca Raton, Ann Arbor, 1994
9. Nakamura, S.: Marching grid generation using parabolic partial differential equations. In: Numerical Grid generation, Thompson, J,.F. (ed.) (1982) 775–786
10. Semenov, A.Yu.: Marching noniterative generation of orthogonal grids. In: Grid Generation: New Trends and Applications in Real-World Simulations, Ivanenko, S.A. and Garanzha, V.A. (eds.), Proc. Minisymp. in Int. Conf. Optimiz. Finite-Element Approximations, Splines and Wavelets, June 25–29, 2001, St. Petersburg, published by Computing Centre of the Russian Academy of Sciences, Moscow (2001) 100–114
11. Steger, J.L. and Chaussee, D.S.: Generation of body-fitted coordinates using hyperbolic partial differential equations, SIAM J. Sci. Stat. Comput. **1** (1980) 431–437
12. Thompson, J.F., Soni, B.K., Weatherill, N.P. (eds.): Handbook of Grid Generation. CRC Press, Boca Raton, Ann Arbor, 1999
13. Zenger, C.: Sparse grids. In Parallel Algorithms for Partial Differential Equations, in: Proc. Sixth GAMM-Seminar, Kiel, 1990, Hackbusch, W. (ed.), volume 31 of Notes on Num. Fluid Mech. Vieweg-Verlag, Braunschweig/ Wiesbaden (1991) 241–251

Computer Algebra in Nanosciences: Modeling Electronic States in Quantum Dots

Dmytro Chibisov, Victor Ganzha, Sergey Pankratov, and Christoph Zenger

Institut für Informatik, Technische Universität München,
Garching 85748, Boltzmannstr. 3, Germany
{chibisov, ganzha, pankrato, zenger}@in.tum.de

Abstract. In the present paper we discuss single-electron states in a quantum dot by solving the Schrödinger equation taking into account spatial constraints, in which the confinement is modeled by a spherical potential wall (particle-in-a-sphere model). After the separation of variables we obtain second order ordinary differential equations, so that automatic methods for finding a closed-form solution are needed. We present a symbolic algorithm implemented in Maple based on the method of indeterminate coefficients, which reduces the obtained equations to the well-known differential equations. The latter can be solved in terms of hypergeometric or Bessel functions. The usage of indeterminate coefficients allows one to obtain the solution of the problem equations in terms of control parameters, which can then be choosen according to the purposes of a nanotechological process.

1 Introduction

Nanotechnology is based on creating artificial structures containing a reduced number of atoms in comparison to the bulk materials. The characteristic dimension of a nanostructure may become smaller than the extension of the electronic wavefunction, at least along some directions. In such cases, the energy of the system is quantized (size quantization), and the system's dimensionality may be reduced, which significantly changes many properties of a nanoobject as compared to its bulk-world counterparts. In the classical language, one can say that nanostructures are in general characterized by a very high surface-to-volume ratio, which results in the evanescent separation between bulk and surface phenomena. For instance, when linear dimensions of a nanostructure become smaller than 10 nm, more than 10 per cent of atoms may be regarded as surface-located. As a result, the physical properties of such nanostructures can be rather extraordinary: a nanostructure typically behaves as a large molecule or even as an atom whose electronic spectrum is determined by its size, shape, composition and, in general, interaction with the environment.

During recent years, a number of research groups have started computational projects aimed to simulate nanodevices and nanotechnological components ([7], [8], [9], [10]). Computational modeling in nanotechnology has readily employed molecular dynamics methods or semiclassical methods aimed to find phase trajectories. However, to study electronic and optical properties of nanostructures, fully quantum mechanical calculations are needed. In this paper we attempt to perform such calculations for zero-dimensional nanostructures - quantum dots, using computer algebra methods as a mathematical tool. In particular, we make use of the following symbolic algorithms

- rewriting techniques to perform the change of variables ([2])
- reduction of the problem to a second order ODE by separation of variables ([2])
- solving the second order ODE by the reduction to the hypergeometric equation using the method of indeterminate coefficients

In the recent years a number of interesting approaches to symbolic solution of second order ODEs has been proposed ([1], [5], [6]). As will be described in Section 2, for nanotechnological purposes it is helpful to model the quantum systems with ODEs whose coefficients include certain control parameters. Thus, the methods are needed, which would allow one to express the solution of parametric ODEs in terms of special functions depending e.g. on external technological parameters. Therefore we present in the Section 3.1 an approach based on the method of indeterminate coefficients and symbolic computation, which would help us to express the solution of a second order ODE in terms of hypergeometric functions containing combinations of external parameters. For example, one can try to interactively model the states of an individual electron in a quantum dot using the classes of potential functions defined in terms of some parameters, such as e.g. with the help of the generalized Hulthén potential (see Section 2).

2 Exploring Spatial Constraints on Electronic States in Quantum Dots

The progress of highly precise nanotechnology made it possible to fabricate quantum dots of lateral dimensions of 10 - 100 nanometers and even smaller (down to 3 nm). Quantum dots (QD) can be produced as single objects within a bulk material or arranged in arrays. It is often said that quantum dots may be considered as artificial atoms, with energy level spacings being quite substantial due to small dimensions. The number of mobility electrons determining electronic and optical properties is typically rather small in a quantum dot ($1 - 10^4$) as compared to the bulk material electronic "sea". Therefore, quantities describing electronic and optical properties of a dot exhibit large fluctuations (or oscillations) as functions of applied (gate) voltage. Such oscillations manifest a sequence of resonances between electronic energy levels and the Fermi energy of the attached leads, which is the physical basis for a controllable and operable nanotechnological systems. With the progress of nanotechnology, the spacing Δ between the electronic energy levels tends to grow together with the temperature region ($kT \leq \Delta$) necessary for effective control, which may reach room temperatures. This fact provides realistic expectations for practical application of nanotechnology on a mass scale.

To explore the role of spatial constraints, let us consider a system of electrons confined in a spherical quantum dot with the radius a. We shall not consider electron-hole pair states at first, although for optical response computations excitonic effects may be crucial. The eigenstates and energy eigenvalues for a single electron in the quantum dot are determined from the Schroedinger equation:

$$(\varDelta + k^2)\psi = 0, \qquad k^2 = \frac{2m(U(r,a) - E)}{\hbar^2}, \tag{1}$$

where E is the particle energy, U is the quantum dot potential, m is the particle mass. The latter may be understood as the effective mass, bearing in mind that quantum dots can be fabricated from the crystalline material and within the crystal matrix, so that electrons moving in the crystal lattice lose the properties of a particle in the free space and become quasiparticles subordinated to the translational symmetry of the lattice and described by the Bloch waves. In fact, however, one can assume that the energy zones are not considerably modified by the quantum confinement, so that the effective mass approximation may be still valid in the quantum dot system. In order to explore the spatial constraints, we approximate the quantum dot with different potential functions $U(r,a)$, which will be discussed bellow.

Using Maple command `PDEchangecoords` from the package `DETools`, we can rewrite equation (1) by simple substitution in spherical coordinates:

$$\varDelta = \frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial r} + \frac{L^2}{r^2}, \tag{2}$$

where

$$L^2 = \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\varphi^2}, \tag{3}$$

is operator depending only on (θ, φ). Separating variables in the traditional way

$$\psi(r,\theta,\varphi) = R(r)\varTheta(\theta,\varphi),\ \ R(r) = \frac{\chi(r)}{r}, \tag{4}$$

we obtain the radial equation in the form

$$-\frac{\hbar^2}{2m}\left(\frac{d^2\chi}{dr^2} - \frac{\lambda}{r^2}\chi\right) + U(r,a)\chi = E\chi \tag{5}$$

and the angular momentum equation

$$L^2\varTheta + \lambda\varTheta = 0. \tag{6}$$

It is well-known that physically meaningful solutions of (6) exist only for $\lambda = l(l+1), l = 0,1,2,..$ Thus, the radial equation can be rewritten as follows:

$$-\frac{\hbar^2}{2m}\left(\frac{d^2\chi}{dr^2} - \frac{l(l+1)}{r^2}\chi\right) + U(r,a)\chi = E\chi \tag{7}$$

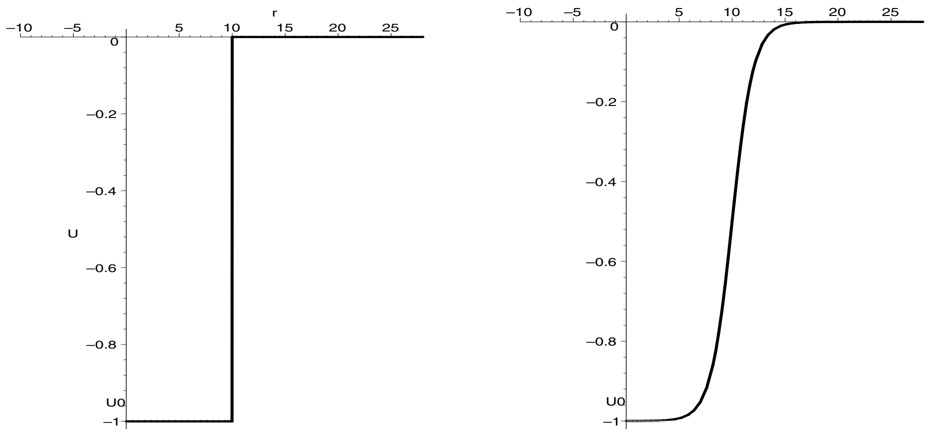


Fig. 1. Potentials in spherical coordinates. Left: hard wall potential, right: Woods-Saxon potential.

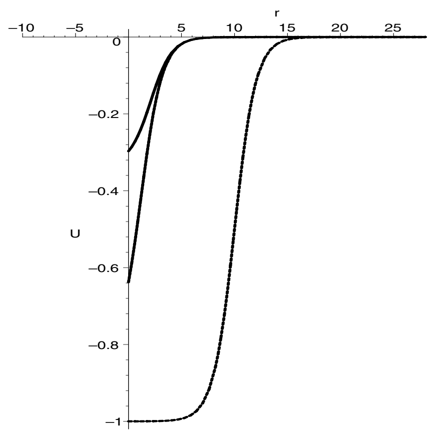


Fig. 2. The generalized Hulthén potential for $\sigma = -1, -1.2, -3$

Solving (7) means finding χ, that is the function of variables r, a, U_0 and E, where $U_0 = max|U((r))|$, i.e. the implicit equation of the form $f(\chi(a, U_0, E)) = 0$ leads to the dispersion relation of the form $E = E(U_0, a)$, thus determining the energy eigenvalues and the respective eigenfunctions in terms of the quantum dot radius and potential. From the nanotechnological point of view, it means that one can control electronic states in a quantum dot by modifying its size and depth. In the next section we shall show how such calculations can be performed using symbolic techniques.

We are interested in a charachterization of the electronic states as a functional of the potential function $U(r, a)$ by solving (7) and imposing the appropriate boundary (supplementary) conditions. This gives us the possibility to explore the role of spatial constraints in the states of electron in a quantum dot. For example, the following simple potential function models a sharp transition between a quantum dot and its environment:

$$U(r,a) = \begin{cases} U_0 & r < a \\ 0 & \text{otherwise} \end{cases}$$

In this case the equation (7) reduces to the following equation inside the dot

$$-\frac{\hbar^2}{2m}\left(\frac{d^2\chi}{dr^2} - \frac{l(l+1)}{r^2}\chi\right) + U_0\chi = E\chi \tag{8}$$

and outside the dot

$$-\frac{\hbar^2}{2m}\left(\frac{d^2\chi}{dr^2} - \frac{l(l+1)}{r^2}\chi\right) = E\chi \tag{9}$$

A slightly more realistic approach to the calculation of particle states in a quantum dot can be based on approximating the latter not by the hard-wall potential, but by that reflecting the finite thickness of the dot surface layer. There may be a number of models taking into account the boundary smear of quantum dots. One of the simplest models of this kind may be based on the Woods-Saxon potential, which was rather popular for constructing nuclear models (and to some extent, by analogy, for models of high-temperature superconductivity in cuprate materials). Another well-used approximation of quantum dot is the generalized Hulthén potential (GHP):

$$U(r,a) = \frac{U_0}{e^{q(r-a)} - \sigma} \tag{10}$$

where σ is a parameter to produce potentials with different characteristics, a may be interpreted as the quantum dot effective radius and the dot is assumed to have a shell with thickness q^{-1}. For large q, the surface layer is very thin and for $q \to \infty$ the quantum dot is described by the hard-wall potential.

By introducing the dimensionless variable

$$z = \frac{1}{e^{q(r-a)} - \sigma}, \tag{11}$$

the radial equation (7) can be transformed to the following form

$$z^2(1-\sigma z)^2\frac{d^2\chi}{dz^2} + [z(1-\sigma z)(1-2\sigma z)]\frac{d\chi}{dz} + \frac{\kappa_0^2 z - \kappa^2}{q^2}\chi = 0, \tag{12}$$

where

$$\kappa^2 = \frac{2m|E|}{h^2},\ \kappa_0^2 = \frac{2mU_0}{h^2}. \tag{13}$$

In the next section, we shall describe a method how equations such as, for example, (8),(9), (12) can be reduced to the hypergeometric equation

$$z(1-z)\phi'' + ((2\beta+1) - 2z(\alpha+\beta+1))\phi' - (\alpha+\beta)(\alpha+\beta+1)\phi = 0 \tag{14}$$

that is solvable in terms of the hypergeometric functions ([3]).

2.1 Symbolic Solution of the Radial Equation Using Maple

In this section, we shall consider how the second order ODE's modelling electronic states can be solved using computer algebra, in particular, with the aid of Maple.Firstly, we consider equations (8) and (12), which can be expressed in terms of Bessel and Hankel functions. After that we shall present an approach based on symbolic computation that allows us to reduce equation (12) to the well-known hypergeometric equation.

Solution of Radial Equations in Terms of Bessel and Hankel Functions. Let us introduce the quantities $\kappa^2 = \frac{2mE}{\hbar^2}$ $\mu^2 = \frac{2m(U_0-E)}{\hbar^2}$. Then the solution of the radial equation inside the dot (8) can be obtained in terms of the Bessel functions:

$$\chi(r) = C_1\sqrt{\frac{\pi\mu r}{2}}J_{l+1/2}(r\mu) + C_2\sqrt{\frac{\pi\mu r}{2}}J_{-(l+1/2)}(r\mu)$$

or, in terms of spherical Bessel functions,

$$j_l(z) = \left(\frac{\pi z}{2}\right)^{1/2} J_{l+1/2}(z),\ n_l(z) = (-1)^{l+1}\left(\frac{\pi z}{2}\right)^{1/2} J_{-(l+1/2)}(z), \tag{15}$$

the radial solution of the Schroedinger equation inside the quantum dot reads:

$$R(r) = \frac{C_1}{r}j_l(\mu r) \tag{16}$$

The spherical function $n_l(\mu r)$ leads to a singularity of $R(r)$ and, hence, must be disregarded. Here the spherical functions are defined in such a way, as to produce the standing plane wave asyptotic values for $z = \mu r \gg l$:

$$j_l(z) \approx sin(z - \frac{\pi l}{2}),\ n_l(z) \approx -cos(z - \frac{\pi l}{2})$$

and the following expansions near $r = 0$, i.e. for $z = \mu r \ll l + 1/2$:

$$j_l(z) \approx \frac{2^l l!}{(2l+1)!}z^{l+1},\ n_l(z) \approx -\frac{(2l)!}{2^l l!}z^{-l}.$$

We are primarily interested in the confined states with $E < 0$. In this case, the only solution of equation (9) outside the dot that is disappearing for $r \gg a$ is the one corresponding to the spherical Hankel function $h_l^{(1)}(z)$ with imaginary z

$$\chi_l(r) = C_2 h_l^{(1)}(i\kappa r),\ r > a,\ i.e.\ R(r) = \frac{C_2}{r}h_l^{(1)}(\kappa r), \tag{17}$$

The constants $C1$ and $C2$ can be determined by the normalization condition together with that stemming from matching the logarithmic derivatives (continuity/smoothness requirement) at r = a, which gives the size quantization condition in the quantum dot:

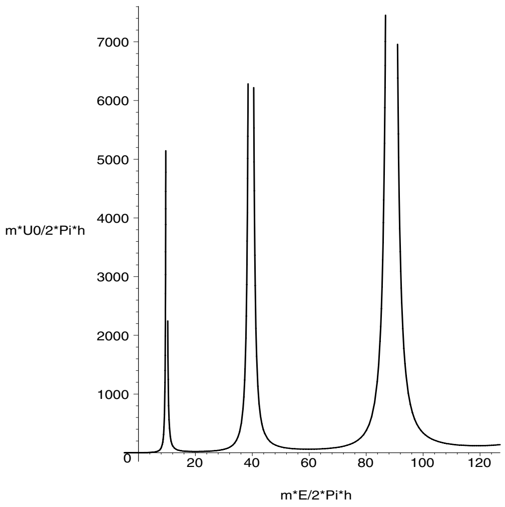


Fig. 3. Solution of (18) for $l = 0$

$$\frac{\mu a j_l'(\mu a)}{j_l(\mu a)} = i\kappa a \frac{h_l^{(1)'}(i\kappa a)}{h_l^{(1)}(i\kappa a)}. \tag{18}$$

Here the prime symbol denotes the differentiation with respect to the whole argument of spherical Bessel functions. The transcendental equation (18) relates the quantities μ and κ, i.e. the implicit equation of the form $f(\mu,\ \kappa) = 0$ leads to the dispersion relation of the form $E = E(U_0, a)$, thus determining the energy eigenvalues and the respective eigenfunctions in terms of the quantum dot radius and potential. From the nanotechnological point of view, it means that one can control electronic states in a quantum dot by modifying its size and depth. For $l = 1/2, 3/2, ...$, the special functions in the equation (18) reduce to trigonometric functions. For example, for $l = 0$, we obtain the following transcendental equation

$$\frac{amE \cos\left(\sqrt{mE}a\right)}{\sqrt{mE} \sin\left(\sqrt{mE}a\right)} = -\sqrt{mU_0 - mE}a,$$

which expresses the relationship between the energy E and potential U_0 as a function of the quantum dot radius a (Fig. 2).

Symbolic Solution of the Radial Equation in Terms of Hypergeometric Function. In this section we shall present an algorithm, which allows us to solve (7) for more complex potential functions $U(r, a)$ such as, for example, (10). Let us re-write the equation (7) in the following general form with polynomial coefficients:

$$a(r)\frac{d^2}{dr^2}\chi(r) + b(r)\frac{d}{dr}\chi(r) + c(r)\chi(r) = 0 \tag{19}$$

where $a(r), b(r), c(r)$ are polynomials of the form

$$a(r) = (a_1 + a_2 r)^2 (a_3 + a_4 r)^2 ; b(r) = (a_1 + a_2 r)(a_3 + a_4 r)(b_2 + b_3 r);$$
$$c(r) = c_2 r + c_1$$

with indeterminate coefficients $a_1, a_2, a_3, a_4, b_2, b_3, c_1, c_2$. In order to reduce the equation (19) to the hypergeometric one, we prove the following statement:

Proposition 1. *The coordinate transformation given by*

$$\chi(r) = \phi(r)(a_1 + a_2 r)^{\alpha}(a_3 + a_4 r)^{\beta}, \tag{20}$$

reduces equation (19) to the equation

$$\begin{aligned}&(a_1 + a_2 r)(a_3 + a_4 r)\phi'' + \\ &(2((\alpha+\beta) a_4 a_2 + 1/2\, b_3) r + 2\, a_2 a_3 \alpha + b_2 + 2\, a_4 a_1 \beta)\phi' + \\ &(a_2 a_4 \alpha + a_2 a_4 \beta + b_3 - a_2 a_4)(\alpha+\beta)\phi = 0\end{aligned} \tag{21}$$

Proof. Cooridnate transformation (20) applied to (19) leads after some simplifications to :

$$\begin{aligned}&(f_1 r^2 + f_2 r + f_3)\phi + (a_1 + a_2 r)^2 (a_3 + a_4 r)^2 \phi'' + \\ &(2\ ((\beta+\alpha) a_4 a_2 + 1/2\, b_3) r + 2\,\alpha\, a_3 a_2 + b_2 + 2\, a_4 a_1 \beta) \\ &(a_1 + a_2 r)(a_3 + a_4 r)\phi' = 0,\end{aligned} \tag{22}$$

where

$$\begin{aligned}f_1 &= a_2 a_4 (\beta+\alpha)(a_2 a_4 \alpha + a_4 a_2 \beta - a_2 a_4 + b_3), \\ f_2 &= 2\, a_1 \beta\ (\alpha - 1 + \beta) a_2 {a_4}^2 + \\ &\quad \left(b_2 (\beta+\alpha) a_2 + 2\,\alpha\, a_3 (\alpha - 1 + \beta) {a_2}^2 + a_1 \beta\, b_3\right) a_4 + \\ &\quad c_2 + \alpha\, a_3 b_3 a_2, \\ f_3 &= {a_1}^2 \beta\ (\beta - 1) {a_4}^2 + 2\, a_1 \beta\ (1/2\, b_2 + \alpha\, a_3 a_2) a_4 + \alpha\, {a_3}^2 (\alpha - 1) {a_2}^2 + \\ &\quad \alpha\, a_3 b_2 a_2 + c_1.\end{aligned}$$

Obviously, equation (22) can be reduced to (21) if the coefficient at ϕ in (22) can be written in the following form :

$$f_1 r^2 + f_2 r + f_3 = (a_1 + a_2 r)(a_3 + a_4 r)(a_2 a_4 \alpha + a_4 a_2 \beta - a_2 a_4 + b_3)(\beta+\alpha) \tag{23}$$

This can be done by expanding the term on the right hand side and collecting coefficients by powers of r:

$$(a_1 + a_2 r)(a_3 + a_4 r)(a_2 a_4 \alpha + a_4 a_2 \beta - a_2 a_4 + b_3)(\beta+\alpha) = g_1 r^2 + g_2 r + g_3$$

where

$$\begin{aligned}g_1 &= a_2 a_4 (\beta+\alpha)(a_2 a_4 \alpha + a_4 a_2 \beta - a_2 a_4 + b_3), \\ g_2 &= (a_1 a_4 + a_2 a_3)(a_2 a_4 \alpha + a_4 a_2 \beta - a_2 a_4 + b_3)(\beta+\alpha), \\ g_3 &= a_1 a_3 (a_2 a_4 \alpha + a_4 a_2 \beta - a_2 a_4 + b_3)(\beta+\alpha).\end{aligned}$$

Since $f_1 = g_1$, the necessary condition for the factorisation (23) is the existence of a solution of the system of equations $\{f_2 = g_2, f_3 = g_3\}$. Using Maple, we obtain the conditions on α and β

$$\beta = \frac{RootOf\left(a_4 c_1 - a_3 c_2 + \left(-a_1 {a_4}^2 + a_4 b_2 + a_3 a_2 a_4 - a_3 b_3\right) Z + a_4 Z^2\right)}{-a_2 a_3 + a_1 a_4} \tag{24}$$

$$\alpha = \frac{RootOf\left(-a_1c_2 + a_2c_1 + \left(-a_2a_1a_4 + a_1b_3 - b_2a_2 + {a_2}^2a_3\right)Z + Z^2a_2\right)}{-a_2a_3 + a_1a_4} \tag{25}$$

where $RootOf(f(Z))$ denotes the complex roots of a polynomial $f(Z)$. In this way equation (19) can be reduced to (21) for α, β given by the above conditions. ♢

Using this result, we can solve equation (12) depending on the potential parameter σ. In the particular case $\sigma = 1$, coefficients in (19) become $a_1 = 0, a_2 = 1, a_3 = 1, a_4 = -1, b_2 = 1, b_3 = -2, c_1 = k_0^2/q^2, c_2 = k^2/q^2$. Equation (21) is then reduced to :

$$z(1-z)\phi'' + ((2\alpha+1) - 2z(\alpha+\beta+1))\phi' - (\alpha+\beta)(\alpha+\beta+1)\phi = 0,$$

where α, β can be obtained by solving quadratic equations (24), (25) symbolically. The obtained expressions are too large to be reproduced here. The complete solution depends on σ and is given by

$$\begin{aligned}\phi(r) = {}&\_C1\ hypergeom\left([\alpha+\beta, \alpha+\beta+1], [1+2\beta], 1-r\right) + \\ &\_C2\ (1-r)^{-2\beta}\ hypergeom\left([\alpha-\beta, -\beta+\alpha+1], [1-2\beta], 1-r\right),\end{aligned}$$

where $hypergeom([a,b],[c],r)$ in Maple notation denotes hypergeometric function ([3]).

3 Conclusion and Outlook

In the present paper a number of symbolic tehchniques has been proposed, which allows one to model quantum dots by solving the second order ODEs using symbolic methods. With regard to nanotechnological purposes, such technique allows one to explore the impact of spatial constraints on electronic states. In this way, one can control electronic states in a quantum dot by modifying its size and depth, in particular by modeling this process interactively. The symbolic method implemented in Maple reduces the model equations to the well-known differential equations. The latter can be solved in terms of hypergeometric or Bessel functions. The solutions obtained by the described techniques can be readily used to compute the electromagnetic (optical) response of quantum dots, with the corresponding integration also being performed by making use of computer algebra methods.

References

1. Bronstein M., Lafaille S.: Solutions of linear ordinary differential equations in terms of special functions, Proceedings of the 2002 international symposium on Symbolic and algebraic computation, July 07-10, 2002, Lille, France, 23–28
2. Cheb-Terrab, E.S., von Blow K.: A Computational approach for the analytical solving of partial differential equations, Computer Physics Communications, 90, 1995, 102–116.

3. Erdelyi, A., Magnus, W., Oberhettinger, F., Tricomi, F. G.: Higher Transcendental Functions, New York: McGraw-Hill Book Company, Inc., 1953
4. A. Gonis: Theory, Modeling, and Computation in Materials Science, LLNL, Livermore, CA, 1993
5. Kovacic J.J. : An algorithm for solving second order linear homogenous differential equations. J. Symbolic Computation, 1986, 2(1), 3–43 (1986) 2
6. Singer M.F. : Liouvillian solutions of linear differential equations with Liouvillian coefficients. J. Symbolic Computation, 1991, 11(3), 251–273 (1991) 11
7. Theory and modeling in nanoscience. Report of the May 10-11, 2002 Workshop, DOE U.S. LBNL-50954
8. Theory, simulation, and modeling in nanoscience, LLNL Nanoscience Home Page http://www.llnl.gov/nanoscience
9. Yoffe, A.D.: Low-dimensional systems: quantum size effects and electronic properties of semiconductor microcristallities (zero-dimensional systems) and some quasi-two-dimensional systems, Adv. Physics, Vol. 51, 2002
10. Yoffe, A.D.: Semiconductor quantum dots and related systems: electronic, optical, luminescence and related properties of lowdimensional systems, Adv. Physics, Vol. 50, 2001

Cayley-Dixon Resultant Matrices of Multi-univariate Composed Polynomials

Arthur D. Chtcherba[1,*], Deepak Kapur[2,*], and Manfred Minimair[3,*,⋆]

[1] University of Texas – Pan American, Dept. of Computer Science,
Edinburg TX 78541, USA
cherba@cs.panam.edu

[2] University of New Mexico, Dept. of Computer Science,
Albuquerque NM 87131, USA
kapur@cs.unm.edu

[3] Seton Hall University, Dept. of Mathematics and Computer Science,
South Orange NJ, USA
manfred@minimair.org

Abstract. The behavior of the Cayley-Dixon resultant construction and the structure of Dixon matrices are analyzed for composed polynomial systems constructed from a multivariate system in which each variable is substituted by a univariate polynomial in a distinct variable. It is shown that a Dixon projection operator (a multiple of the resultant) of the composed system can be expressed as a power of the resultant of the *outer* polynomial system multiplied by powers of the leading coefficients of the univariate polynomials substituted for variables in the outer system. The derivation of the resultant formula for the composed system unifies all the known related results in the literature. A new resultant formula is derived for systems where it is known that the Cayley-Dixon construction does not contain any extraneous factors. The approach demonstrates that the resultant of a composed system can be effectively calculated by considering only the resultant of the outer system.

1 Introduction

Problems in many application domains, including engineering and design, graphics, CAD-CAM, geometric modeling, etc. can be modelled using polynomial systems [1–8]. Often a polynomial system arising from an application has a structure. Particularly in engineering and design applications and in geometric modeling, a polynomial system can be expressed as a composition of two distinct polynomial systems, each of which is of much lower degree in comparison to the original system. Furthermore, if the structure of given polynomials is not known a priori, one can efficiently check if they can be decomposed [9].

* Supported by NSF grant no. CCR-0203051 and a grant from the Computer Science Research Institute at the Sandia National Labs.

⋆ Also supported by NSF grant CCF 0430741.

This paper addresses the resultant computation for such composed polynomial systems [10–14]. The resultant of a polynomial system with symbolic parameters is a necessary and sufficient condition on its parameters for the polynomial system to have a common solution[1]. Resultant computations have been found useful in many application domains including engineering and design, robotics, inverse kinematics, manufacturing, design and analysis of nano devices in nanotechnology, image understanding, graphics, solid modeling, implicitization, CAD-CAM design, geometric construction, drug-design, and control theory.

The focus in this paper is on the Cayley-Dixon formulation for multivariate resultants which have been shown to be efficient (both experimentally and theoretically) for computing resultants by simultaneously eliminating many variables from a polynomial system [16]. The behavior of the Cayley-Dixon resultant construction is analyzed for composed polynomial systems constructed from a multivariate system in which each variable is substituted by a univariate polynomial in a distinct variable, referred to as multi-univariate composition in [9]. It is shown that the resultant of the composed system can be expressed as a power of the resultant of the *outer* polynomial system, multiplied by powers of the leading coefficients of the univariate polynomials substituted for variables in the outer system. It is important to point out that the techniques used for deriving resultant formulas in the current paper are different from the techniques used in previous works (such as [10–13,17,18]), which seemed not applicable.

A new resultant formula is derived for multi-univariate composed polynomials where it is known that the Cayley-Dixon resultant formulation does not produce any extraneous factors for the outer system. The derivation unifies all known related results in the literature [18,19]. Such systems include n-degree [8], bivariate corner cut [20] and generalized corner cut systems [21]. Even when extraneous factors are present, a similar formula is derived showing that the extraneous factor of the outer system will be "amplified" in the extraneous factor of composed system. Hence exploiting the composed structure of a polynomial system can reduce the extraneous factors in the resultant computation. Furthermore, it demonstrates that the resultant of a composed system can be effectively calculated by considering only the resultant of the outer system. For practical applications, that is what is needed.

Below, we first state the main result of the paper. This is followed by a section on preliminaries and notation; the generalized Cayley-Dixon formulation as proposed by Kapur, Saxena and Yang [8] is briefly reviewed. Since the Cayley-Dixon formulation involves two disjoint sets of variables, the bilinear form representation of a polynomial in disjoint sets of variables is useful. In section 2, we discuss how bilinear forms are affected by polynomial operations, particularly when two polynomials are multiplied, a polynomial is composed with other polynomials by substituting variables by polynomials etc. To express these relations among bilinear forms, a series of matrix operations is introduced.

We assume that the reader is familiar with the notion of resultant with respect to a given variety (see for example [15]). This notion includes classic resultants

[1] Resultant depends on an algebraic set for which solutions are sought [15].

like the projective (Macaulay) resultant where the variety is projective space and more recent generalizations like toric resultants where the varieties are suitable toric varieties.

1.1 Main Results

Consider a polynomial system $F = (f_0, f_1, \ldots, f_n)$ with symbolic coefficients, where $F \subset \mathbb{K}[\mathbf{c}][y_1, \ldots, y_n]$ and

$$f_i = \sum_{\alpha \in \mathcal{F}_i} c_{i,\alpha} \mathbf{y}^\alpha \qquad \text{for} \quad i = 0, \ldots, n,$$

where $\mathbf{y}^\alpha = y_1^{\alpha_1}, \ldots, y_n^{\alpha_n}$ and $\mathcal{F}_i$ is the set of exponent vectors corresponding to the terms appearing in f_i, also called the *support* of f_i. The list $\mathbf{c}$ consists of "other" variables in terms of which polynomial coefficients $c_{i,\alpha} \in \mathbb{K}[\mathbf{c}]$ are defined. They are also sometimes referred as the *parameters* of the polynomial system.

A polynomial system is called *generic* if there is no algebraic relation among coefficients $c_{i,\alpha}$ of F.

Let $G = (g_1, \ldots, g_n)$ be a *univariate* polynomial system where

$$g_j(x_j) = d_{j,k_j} x_j^{k_j} + d_{j,k_j-1} x_j^{k_j-1} + \cdots + d_{j,0}, \quad \text{for} \quad j = 1, \ldots, n.$$

Let $k = (k_1, \ldots, k_n)$ be the degree vector of G.

We consider **composed** polynomial system of F with G, written as $F \circ G$, which is the list of polynomials obtained from the list F of polynomials by replacing each y_j by g_j respectively. The operator $\circ$ is called *functional composition* on polynomial systems.

The main results of this paper are:

(i) The Dixon matrix $\Theta_{F \circ G}$ of a composed system $F \circ G$ is shown to be a product of 3 matrices:

$$\Theta_{F \circ G} = A_{\mathrm{L}} \times \mathrm{Diag}_{k_1 \cdots k_n}(\Theta_F) \times A_{\mathrm{R}},$$

where Θ_F is the Dixon matrix of the outer system F and A_{L} as well as A_{R} are triangular matrices which contain only polynomials in terms of the coefficients of the polynomials in G. The matrix $\mathrm{Diag}_{k_1 \cdots k_n}(\Theta_F)$ is block diagonal, where Θ_F is repeated $k_1 \cdots k_n$ times along the diagonal.

(ii) If F is a polynomial system for which the determinant of Dixon matrix is $\mathrm{Res}(F)$, then

$$\mathrm{Res}(F \circ G) = d_{1,k_1}^{\epsilon_1} \cdots d_{n,k_n}^{\epsilon_n} \mathrm{Res}(F)^\delta,$$

where ϵ_j's depend on the degrees of G as well as F but δ depends only on the degrees of G.

(iii) Even if Θ_F is not square or is singular, the *rank submatrix construction* (RSC) introduced in [8] (see also [15]) also works for composed systems. In particular, the projection operator extracted from Θ_F is a factor of the projection operator extracted from $\Theta_{F \circ G}$ raised to the appropriate power; in addition to the leading coefficients d_{j,k_j} of polynomials in G, there are also additional factors introduced in the projection operator extracted from $\Theta_{F \circ G}$.

(iv) The resultant of composed n-degree system, with degrees $(m_1, \ldots, m_n)$, is

$$\operatorname{Res}(F \circ G) = \left(d_{1,k_1}^{m_1} \cdots d_{n,k_n}^{m_n}\right)^{\frac{(n+1)!}{2}\, m_1 \cdots m_2\, k_1 \cdots k_n} \operatorname{Res}(F)^{k_1 \cdots k_n}.$$

2 Cayley-Dixon Formulation and Bilinear Form

2.1 The Cayley-Dixon Formulation

In [22], Dixon extended the Bezout-Cayley's construction for computing the resultant of two univariate polynomials to the bivariate case for three polynomials. Kapur, Saxena and Yang [8] generalized this construction to the multivariate case. The concepts of a Dixon polynomial and a Dixon matrix were introduced. Below, the generalized multivariate Dixon formulation for simultaneously eliminating many variables from a polynomial system and computing its resultant are briefly reviewed. Let $\pi_i(\mathbf{y}^\alpha) = \overline{y}_1^{\alpha_1} \cdots \overline{y}_i^{\alpha_i} y_{i+1}^{\alpha_{i+1}} \cdots y_n^{\alpha_n}$, where $i \in \{0, 1, \ldots, n\}$, and $\overline{y}_i$'s are new variables; $\pi_0(\mathbf{y}^\alpha) = \mathbf{y}^\alpha$. π_i is extended to polynomials in a natural way as: $\pi_i(f_j(y_1, \ldots, y_n)) = f_j(\overline{y}_1, \ldots, \overline{y}_i, y_{i+1}, \ldots, y_n)$.

Definition 1. *Given a n-variate polynomial system $F = (f_0, f_1, \ldots, f_n)$, where $f \subset \mathbb{K}[\mathbf{c}][y_1, \ldots, y_n]$, define its* ***Dixon polynomial*** *as*

$$\theta(F) = \prod_{i=1}^{n} \frac{1}{\overline{y}_i - y_i} \det \begin{pmatrix} \pi_0(f_0) & \pi_0(f_1) & \cdots & \pi_0(f_n) \\ \pi_1(f_0) & \pi_1(f_1) & \cdots & \pi_1(f_n) \\ \vdots & \vdots & \ddots & \vdots \\ \pi_n(f_0) & \pi_n(f_1) & \cdots & \pi_n(f_n) \end{pmatrix} = \overline{Y}^T \times \Theta_F \times Y,$$

where $\overline{Y} = [\overline{\mathbf{y}}^{\beta_1}, \ldots, \overline{\mathbf{y}}^{\beta_k}]$ and $Y = [\mathbf{y}^{\alpha_1}, \ldots, \mathbf{y}^{\alpha_l}]$ are column vectors. Hence, $\theta(f_0, f_1, \ldots, f_n) \in \mathbb{K}[\mathbf{c}][y_1, \ldots, y_n, \overline{y}_1, \ldots, \overline{y}_n]$, where $\overline{y}_1, \ldots, \overline{y}_n$ are new variables. The $k \times l$ matrix Θ_F is called the ***Dixon matrix****, and its entries are in $\mathbb{K}[\mathbf{c}]$.*

The order in which original variables in $\mathbf{y}$ are replaced by new variables in $\overline{\mathbf{y}}$ is significant in the sense that the computed Dixon polynomial can be different for two different orderings. See [22,8,21,15].

As shown in [8] and [15], Θ_F is a resultant matrix, i.e., the resultant can be computed from the determinant of Θ_F. If Θ_F singular, for example for certain nongeneric polynomial systems, then the resultant is extracted from the determinant of some maximal minor of Θ_F; this determinant is called a *projection operator* [8,15].

2.2 Operations on Bilinear Forms

It is easy to see that a multivariate polynomial in terms of two disjoint sets of variables, e.g., the Dixon polynomial above, can be represented in a *bilinear form*. For analyzing how the functional composition of two polynomial systems affects the Dixon polynomials and Dixon matrices of the polynomial systems, bilinear form representations turn out to be useful. Below, we discuss various polynomial operations and their effect on bilinear forms.

A bilinear form of a polynomial p in two disjoint sets of variables is expressed as a matrix, post and pre-multiplied by monomial vectors. That is

$$p(x_1, \dots, x_k, \overline{x}_1, \dots, \overline{x}_l) = \sum_{\alpha,\beta} p_{\alpha,\beta}\, \mathbf{x}^{\alpha}\overline{\mathbf{x}}^{\beta} \;=\; \overline{X_p}^{\,T} \times M_p \times X_p,$$

where $\overline{X_p}$ and X_p are vectors with entries being monomials in terms of variables $\{\overline{x}_1, \dots, \overline{x}_l\}$ and $\{x_1, \dots, x_k\}$, respectively. M_p is a matrix with the coefficients $p_{\alpha,\beta}$ of terms in p as its entries.

The matrix M_p in the above definition depends on the monomial ordering used. We will assume a total degree ordering on power products, and state explicitly if it is otherwise. Also, implicit in the above definition of M_p are the row labels $\overline{X}_p$ and column labels X_p.

Let $\mathcal{P}$ be the ordered set of the exponent vectors corresponding to X_p; $\mathcal{P}$ is also called the **support** of the polynomial p w.r.t variables $\{x_1, \dots, x_k\}$. Similarly, let $\overline{\mathcal{P}}$ be the support of p w.r.t. variables $\{\overline{x}_1, \dots, \overline{x}_l\}$. Let $\mathcal{P} + \mathcal{Q}$ stand for the Minkowski sum of the supports $\mathcal{P}$ and $\mathcal{Q}$. [23]

As stated above, the Dixon polynomial can be conveniently represented in bilinear form using the original variables and the new variables, highlighting the Dixon matrix. Let Δ_F and $\overline{\Delta}_F$ be the supports of the Dixon polynomial $\theta(F)$ in terms of variables $\mathbf{y}$ and $\overline{\mathbf{y}}$, respectively.

Below, we derive the bilinear matrix form of the product of two polynomials in terms of their bilinear matrix forms. For this purpose, we first define the so-called "left" and "right" operators on bilinear forms.

Definition 2. *Given two polynomials p and q admitting bilinear form, i.e, $p = \overline{X}_p \times M_p \times X_p$ and $q = \overline{X}_q \times M_q \times X_q$, consider the following polynomial products*

$$\begin{aligned} p' &= p \cdot \sum_{\overline{e}_q \in \overline{\mathcal{Q}}} \mathbf{z}^{\overline{e}_q}\overline{\mathbf{x}}^{\overline{e}_q} = \overline{X}_{p'} \times M_{p'} \times X_{p'}, \\ q' &= q \cdot \sum_{e_p \in \mathcal{P}} \overline{\mathbf{z}}^{e_p}\mathbf{x}^{e_p} = \overline{X}_{q'} \times M_{q'} \times X_{q'}, \end{aligned} \qquad \textit{and}$$

where $\overline{z}_1, \dots, \overline{z}_n$ and $z_1, \dots, z_n$ are new variables. Define two matrix operators

$$\mathrm{L}_{\overline{\mathcal{Q}}}(M_p) = M_{p'} \qquad \textit{and} \qquad \mathrm{R}_{\mathcal{P}}(M_q) = M_{q'},$$

where columns of $M_{p'}$ ordered first by some monomial order on $\{z_1, \dots, z_n\}$ and then by some monomial order on $\{x_1, \dots, x_n\}$. Similarly rows of $M_{q'}$ first *ordered by $\{\overline{x}_1, \ldots, \overline{x}_n\}$ and then by $\{\overline{z}_1, \ldots, \overline{z}_n\}$. To make matrices $M_{p'}$ and $M_{q'}$ "compatible" with each other, we require that monomial order used for columns of $M_{p'}$ on variables $[\{z_1, \ldots, z_n\}, \{x_1, \ldots, x_n\}]$ be same as for rows of M'_q on variables $[\{\overline{x}_1, \ldots, \overline{x}_n\}, \{\overline{z}_1, \ldots, \overline{z}_n\}]$.*

The above matrix operators are defined in such a way that matrix multiplication would coincide with polynomial multiplication. New variables $\overline{z}_1, \ldots, \overline{z}_n$ and $z_1, \ldots, z_n$ are auxiliary variables for creating block matrix structure, as well as ensuring that the resulting matrix rows and columns are in matching order. Notice that the row indices of $\mathrm{L}_{\overline{\mathcal{Q}}}(M_p)$ are $\overline{\mathcal{P}} + \overline{\mathcal{Q}}$ and the column indices are $\overline{\mathcal{Q}} \times \mathcal{P}$, coming from monomials $\mathbf{z}^{\overline{e}_q}\mathbf{x}^{e_p}$ for $\overline{e_q} \in \overline{\mathcal{Q}}$ and $e_p \in \mathcal{P}$.

Matrix $\mathrm{L}_{\overline{\mathcal{Q}}}(M_p)$ is quite sparse and its entries are either 0 or the coefficients of the polynomial p. In fact, the entry of $\mathrm{L}_{\overline{\mathcal{Q}}}(M_p)$ indexed by row $\overline{\mathbf{x}}^{\overline{e}_p+\overline{e}_q}$ and column $\mathbf{z}^{\overline{e}_q}\mathbf{x}^{e_p+e_q}$ is equal to $p_{\overline{e}_p,e_p}$. All other entries are 0. Also it has a block matrix structure: $\mathrm{L}_{\overline{\mathcal{Q}}}(M_p) = \mathrm{RowStack}_{\alpha\in\overline{\mathcal{Q}}}(N_\alpha \times M_p)$, where N_α is a matrix which adds zero rows to M_p (depending on α, $\overline{\mathcal{Q}}$ and $\overline{\mathcal{P}}$), and operator RowStack stacks matrices columnwise. $\mathrm{R}_{\mathcal{P}}(M_q)$ also admits a similar block decomposition.

Using the above operators, we can express bilinear form of the polynomial product, $pq = \overline{X}_{pq} \times M_{pq} \times X_{pq}$ as matrix multiplication.

Lemma 1. $$M_{p\,q} = \mathrm{L}_{\overline{\mathcal{Q}}}(M_p) \times \mathrm{R}_{\mathcal{P}}(M_q)\,.$$

Proof. Directly from the polynomial product of polynomials p and q,

$$(M_{p\,q})_{\alpha,\beta} = \sum_{\substack{\alpha=\overline{e}_p+\overline{e}_q,\\ \beta=e_p+e_q}} p_{\overline{e}_p,e_p} q_{\overline{e}_q,e_q},$$

for $\overline{e}_p \in \overline{\mathcal{P}}$, $\overline{e}_q \in \overline{\mathcal{Q}}$, $e_p \in \mathcal{P}$ and $e_q \in \mathcal{Q}$. On the other hand,

$$\begin{aligned}\left(\mathrm{L}_{\overline{\mathcal{Q}}}(M_p) \times \mathrm{R}_{\mathcal{P}}(M_q)\right)_{\alpha,\beta} &= \mathrm{Row}_\alpha\left(\mathrm{L}_{\overline{\mathcal{Q}}}(M_p)\right) \cdot \mathrm{Col}_\beta\left(\mathrm{R}_{\mathcal{P}}(M_q)\right)\\ &= \sum_{\substack{\overline{e}_q\in\overline{\mathcal{Q}},\\ e_p\in\mathcal{P}}} p'_{\alpha,\overline{e}_q e_p} \cdot q'_{\overline{e}_q e_p,\beta},\end{aligned}$$

where $p'_{\alpha,\overline{e}_q e_p}$ is the coefficient of monomial $\overline{\mathbf{x}}^\alpha \mathbf{z}^{\overline{e}_q}\mathbf{x}^{e_p}$ of p'. But

$$p'_{\alpha,\overline{e}_q e_p} = \begin{cases} p_{\overline{e}_p,e_p} & \text{if } \alpha = \overline{e}_p + \overline{e}_q \\ 0 & \text{otherwise}\end{cases}, \quad \text{and} \quad q'_{e_p\overline{e}_q,\beta} = \begin{cases} q_{\overline{e}_q,e_q} & \text{if } \beta = e_p + e_q \\ 0 & \text{otherwise}\end{cases}.$$

where $e_p \in \mathcal{P}$, $\overline{e}_p \in \overline{\mathcal{P}}$, $q \in \overline{\mathcal{Q}}$ and $\overline{e}_q \in \overline{\mathcal{Q}}$. Therefore

$$\sum_{\substack{\overline{e}_q\in\overline{\mathcal{Q}},\\ e_p\in\mathcal{P}}} p'_{\alpha,\overline{e}_q e_p} \cdot q'_{\overline{e}_q e_p,\beta} = \sum_{\substack{\alpha=\overline{e}_p+\overline{e}_q,\\ \beta=e_p+e_q}} p'_{\alpha,\overline{e}_q e_p} \cdot q'_{\overline{e}_q e_p,\beta} = \sum_{\substack{\alpha=\overline{e}_p+\overline{e}_q,\\ \beta=e_p+e_q}} p_{\overline{e}_p,e_p} q_{\overline{e}_q,e_q} .$$ □

One of the useful properties of L operator is that the application on matrix product results in the application on one of the matrices times a block diagonal matrix of the other factor.

Lemma 2. *Given a product of two matrices $A \times B$,*

$$\mathrm{L}_{\mathcal{P}}(A \times B) = \mathrm{L}_{\mathcal{P}}(A) \times \mathrm{Diag}_{\mathcal{P}}(B)\,,$$

where matrix $\mathrm{Diag}_{\mathcal{P}}(B)$ *is block diagonal with B repeated $|\mathcal{P}|$ times along the diagonal.*

Proof. By definition,

$$\begin{aligned}\mathrm{L}_{\mathcal{P}}(A \times B) &= \mathrm{RowStack}_{\alpha\in\mathcal{P}}(N_\alpha \times (A \times B)) = \mathrm{RowStack}_{\alpha\in\mathcal{P}}((N_\alpha \times A) \times B)\\ &= \mathrm{RowStack}_{\alpha\in\mathcal{P}}(N_\alpha \times A) \times \mathrm{Diag}_{\mathcal{P}}(B) = \mathrm{L}_{\mathcal{P}}(A) \times \mathrm{Diag}_{\mathcal{P}}(B)\,. \qquad \square\end{aligned}$$

Note that if $|\overline{\mathcal{P}} + \overline{\mathcal{Q}}| = |\overline{\mathcal{P}}| \times |\overline{\mathcal{Q}}|$, for example when p and q are in terms of different variables, then $\mathrm{L}_{\overline{\mathcal{Q}}}(M_p) = \mathrm{Diag}_{\overline{\mathcal{Q}}}(M_p)$.

Definition 3. *Given a support $\mathcal{P}$ and the set of univariate polynomials $G = (g_1, \ldots, g_n)$, where each g_i is in x_i, let*

$$s = \sum_{\alpha\in\mathcal{P}} \overline{\mathbf{x}}^{\alpha} G^{\alpha} = \overline{X}_s \times M_s \times X_s,$$

where $G^\alpha = \prod_{i=1}^n g_i^{\alpha_i}$. Define operator $\mathrm{S}_{\mathcal{P}}(G) = M_s$.

$\mathrm{S}_{\mathcal{P}}(G)$ is thus the matrix whose rows are indexed by $\mathcal{P}$ and whose columns are indexed by the union over $\alpha \in \mathcal{P}$ of the supports of $\prod_{j=1}^n g_j^{\alpha_j}$. Note that the monomial vector, with support $\mathcal{P}$ composed with G can be expressed as $Y_p \circ G = \mathrm{S}_{\mathcal{P}}(G) \times X_s$, where X_s is union of all monomials in G^α for all $\alpha \in \mathcal{P}$. Matrix $\mathrm{S}_{\mathcal{P}}(G)$ is also very sparse and it is "step"-triangular (i.e., where in each row, first non-zero entry comes later than in the previous row), i.e.,

$$(\mathrm{S}_{\overline{\mathcal{P}}}(G))_{\overline{e}_s, e_s} = \begin{cases} (d_{1,k_1}, \ldots, d_{n,k_n})^{\overline{e}_s} & \text{if } (e_s)_i = k_i(\overline{e}_s)_i,\ \forall i,\\ 0 & \text{if } \exists\, i \text{ s.t. } k_i(\overline{e}_s)_i < (e_s)_i. \end{cases} \tag{1}$$

Lemma 3. *Let p be a polynomial in the variables $\overline{\mathbf{y}}$, $\mathbf{y}$, and G a set of univariate polynomials g_i in variable x_i, for $i = 1, \ldots, n$. Then*

$$M_{p\,\circ\,(\overline{G},\,G)} \;=\; \mathrm{S}_{\overline{\mathcal{P}}}(\overline{G})^{\mathrm{T}} \times M_p \times \mathrm{S}_{\mathcal{P}}(G)\,,$$

where $\overline{G} = (\overline{g}_1, \ldots, \overline{g}_n)$, and $\overline{g}_i = g_i(\overline{x}_i)$.

Proof. Since $p = \overline{Y}_p \times M_p \times Y_p$, we have $p \circ (\overline{G}, G) \;=\; (\overline{Y}_p^T \circ \overline{G}) \times M_p \times (Y_p \circ G)$ and $Y_p \circ G = \mathrm{S}_{\mathcal{P}}(G) \times X_s$ by definition. $\square$

A very useful property of operators L and S is that in combination, they produce step-triangular matrices[2]. Square step-triangular matrices are triangular.

[2] See the expanded version of this article [24] for many examples illustrating the structure of of these matrices.

Proposition 1. *Let $\overline{\mathcal{Q}}$ be the support of $\prod_{i=1}^{n} \frac{g_i - \overline{g}_i}{x_i - \overline{x}_i}$, that is $\overline{e}_q \in \overline{\mathcal{Q}}$ iff $0 \le (\overline{e}_q)_i < k_i$ for all $i = 1, \ldots, n$. Then for any support $\mathcal{P}$, the matrix $\mathrm{L}_{\overline{\mathcal{Q}}}\Big(\mathrm{S}_{\mathcal{P}}\big(\overline{G}\big)^{\mathrm{T}}\Big)$ is (after column reordering) is zero above the step diagonal; moreover, entry in column $\overline{e}_q e_p$ (i.e. indexed by monomial $\mathbf{z}^{\overline{e}_q}\mathbf{x}^{e_p}$) and row α is*

$$\mathrm{L}_{\overline{\mathcal{Q}}}\Big(\mathrm{S}_{\mathcal{P}}\big(\overline{G}\big)^{\mathrm{T}}\Big)_{\alpha, \overline{e}_q e_p} = \begin{cases} (d_{1,k_1}, \ldots, d_{n,k_n})^{e_p} & \text{if } \alpha = \overline{e}_p + \overline{e}_q \text{ and } (\overline{e}_p)_i = k_i (e_p)_i, \\ \Big(\mathrm{S}_{\mathcal{P}}\big(\overline{G}\big)^{\mathrm{T}}\Big)_{\overline{e}_p, e_p} & \text{if } \alpha = \overline{e}_p + \overline{e}_q \text{ and } \forall i,\ (\overline{e}_p)_i < k_i (e_p)_i, \\ 0 & \text{otherwise,} \end{cases}$$

i.e., in every column, first non-zero entry is the product of leading coefficients of G, and all these leading non-zero entries are in different rows.

Proof. Note that the columns of $\mathrm{S}_{\mathcal{P}}\big(\overline{G}\big)^{\mathrm{T}}$ are labelled by $\mathcal{P}$ and rows by $\overline{X}_s$, which is the set of all monomials in $\overline{G}^{\alpha} = \overline{g}_1^{\alpha_1} \cdots \overline{g}_n^{\alpha_n}$ for all $\alpha \in \mathcal{P}$.

Consider the following polynomial,

$$s = \overline{X}_s \times \mathrm{S}_{\mathcal{P}}\big(\overline{G}\big)^{\mathrm{T}} \times X_s, \quad \text{and let} \quad s' = s \cdot \sum_{\overline{e}_q \in \overline{\mathcal{Q}}} \mathbf{z}^{\overline{e}_q}\, \overline{\mathbf{x}}^{\overline{e}_q},$$

as in definition 2 of $\mathrm{L}_{\overline{\mathcal{Q}}}(M_p)$. As in the proof of Lemma 1, $\mathrm{coeff}_{\overline{\mathbf{x}}^{\alpha}\mathbf{z}^{\overline{e}_q}\mathbf{x}^{e_s}}(s') = s_{\overline{e}_s, e_s}$ iff $\alpha = \overline{e}_s + \overline{e}_q$, and 0 otherwise. Since the support of s is $\mathcal{P}$, we will use labels e_p instead of e_s. Equation (1) and the above observation gives us

$$\mathrm{coeff}_{\overline{\mathbf{x}}^{\alpha}\mathbf{z}^{\overline{e}_q}\mathbf{x}^{e_p}}(s') = \begin{cases} (d_{1,k_1}, \ldots, d_{n,k_n})^{e_p} & \text{if } \alpha = \overline{e}_s + \overline{e}_q \text{ and } (\overline{e}_s)_i = k_i (e_p)_i, \\ s_{\overline{e}_s, e_p} & \text{if } \alpha = \overline{e}_s + \overline{e}_q \text{ and } \forall i,\ (\overline{e}_s)_i < k_i (e_p)_i, \\ 0 & \text{otherwise.} \end{cases} \qquad \square$$

In the next section, we use the above operators in expressing the manipulations of bilinear forms of various polynomials arising in the Cayley-Dixon construction to show that Dixon matrix of composed system can be decomposed as a matrix product.

3 Dixon Matrix Decomposition

The **Cayley-Dixon Construction** of the composed polynomials $F \circ G$ is a generalization of the Cayley-Bézout construction from the univariate case. The Dixon polynomial of the composed system

$$\begin{aligned} \theta_{F \circ G} &= \frac{\det \begin{bmatrix} f_0 \circ (\pi_0(G)) & \ldots & f_n \circ (\pi_0(G)) \\ \vdots & \ddots & \vdots \\ f_0 \circ (\pi_n(G)) & \ldots & f_n \circ (\pi_n(G)) \end{bmatrix}}{\prod_{i=1}^{n} (g_i - \overline{g}_i)} \times \frac{\prod_{i=1}^{n} (g_i - \overline{g}_i)}{\prod_{i=1}^{n} (x_i - \overline{x}_i)} \\ &= \theta_F \circ (\overline{G}, G) \times \prod_{i=1}^{n} \frac{g_i - \overline{g}_i}{x_i - \overline{x}_i}. \end{aligned} \tag{2}$$

Let $p = \theta_F \circ (\overline{G}, G)$ and $q = \prod_{i=1}^{n} \frac{g_i - \overline{g}_i}{x_i - \overline{x}_i}$, where $\mathcal{P}$ is the support of p with respect to the variables $x_1, \ldots, x_n$ and $\overline{\mathcal{Q}}$ is the support of q with respect to the variables $\overline{x}_1, \ldots, \overline{x}_n$. Using Lemmas 1, 2 and 3 to equation (2) above, we get

$$\Theta_{F \circ G} = \mathrm{L}_{\overline{\mathcal{Q}}}\Big(\mathrm{S}_{\overline{\Delta}_F}(G)^{\mathrm{T}}\Big) \times \mathrm{Diag}_{\overline{\mathcal{Q}}}(\Theta_F) \times \big(\mathrm{Diag}_{\overline{\mathcal{Q}}}(\mathrm{S}_{\Delta_F}(G)) \times \mathrm{R}_{\mathcal{P}}(M_q)\big) .$$

Theorem 1. *For a polynomial system $F = (f_0, f_1, \ldots, f_n)$ and a list of univariate polynomials $G = (g_1, \ldots, g_n)$, the Dixon matrix $\Theta_{F \circ G}$ is*

$$A_{\mathrm{L}} \times \mathrm{Diag}_{k_1 \cdots k_n}(\Theta_F) \times A_{\mathrm{R}},$$

where A_{L} and A_{R} are step triangular matrices with diagonal entries being the product of the leading coefficients of the polynomials in G. Specifically,

$$A_{\mathrm{L}} = \mathrm{L}_{\overline{\mathcal{Q}}}\Big(\mathrm{S}_{\overline{\Delta}_F}(\overline{G})^{\mathrm{T}}\Big), \qquad \textit{and} \qquad A_{\mathrm{R}} = \mathrm{Diag}_{\overline{\mathcal{Q}}}(\mathrm{S}_{\Delta_F}(G)) \times \mathrm{R}_{\mathcal{P}}(\mathrm{M}_q),$$

where $q = \prod_{i=1}^{n} \frac{g_i - \overline{g}_i}{x_i - \overline{x}_i}$ is the product of the divided differences of G, $\mathcal{Q}$ and $\overline{\mathcal{Q}}$ are, respectively, the supports of q in the variables $\mathbf{x}$ and $\overline{\mathbf{x}}$.

More importantly, for a generic n-degree polynomial system F and a generic system G of n polynomials used to substitute for variables $y_1, \cdots, y_n$ in F, the factors, A_{L}, A_{R} and Θ_F can be proved to be square and non-singular matrices [8]. We investigate this in Section 4.

More generally, if the factors are square in the above theorem, then we can derive precise expression for the determinant of the Dixon matrix.

Lemma 4. *If $|\overline{\Delta}_F| \cdot \prod_{j=1}^{n} k_j = |\overline{\Delta}_{F \circ G}|$, i.e., A_{L} is square, then*

$$\det(A_{\mathrm{L}}) = \pm (d_{1,k_1}, \ldots, d_{n,k_n})^{(\sum_{\alpha \in \overline{\Delta}_F} \alpha)\, k_1 \cdots k_n};$$

if $|\Delta_F| = |\overline{\Delta}_F|$, i.e., Θ_F is square, then

$$\det\Big(\mathrm{Diag}_{|\overline{\mathcal{Q}}|}(\Theta_F)\Big) = (\det(\Theta_F))^{k_1 \cdots k_n};$$

and if $|\Delta_F| \cdot \prod_{j=1}^{n} k_j = |\Delta_{F \circ G}|$, i.e., A_{R} is square, then

$$\det(A_{\mathrm{R}}) = \pm (d_{1,k_1}, \ldots, d_{n,k_n})^{(|\Delta_F| + \sum_{\beta \in \Delta_F} \beta) k_1 \cdots k_n} .$$

Proof. When A_L is square, it is triangular with diagonal entries

$$(A_L)_{\alpha, \overline{e}_q \cdot e_p} = (d_{1,k_1}, \ldots, d_{n,k_n})^{e_p}$$

in column $\overline{e}_q e_p$, where $e_p \in \mathcal{P} = \overline{\Delta}_F$, by Proposition 1. Since the size of $\overline{Q}$ is $k_1 \cdots k_n$,

$$\det(A_L) = \prod_{\substack{e_p \in \overline{\Delta}_F \\ \overline{e}_q \in \overline{\mathcal{Q}}}} (d_{1,k_1}, \ldots, d_{n,k_n})^{e_p} = (d_{1,k_1}, \ldots, d_{n,k_n})^{(\sum_{\alpha \in \overline{\Delta}_F} \alpha)\, k_1 \cdots k_n} .$$

Also, for $A_R = \mathrm{Diag}_{\overline{\mathcal{Q}}}(\mathrm{S}_{\Delta_F}(G)) \times \mathrm{R}_{\mathcal{P}}(\mathrm{M}_q)$, let $s = \overline{Z}_s \times \mathrm{S}_{\Delta_F}(G) \times X_s$, $A_R = M_{sq}$, as in the univariate case. Also note that M_q is triangular, where

$$q_{\overline{e}_q, e_q} = \begin{cases} d_{1,k_1} \cdots d_{n,k_n} & \text{if } \forall i \ \text{ s.t. } (\overline{e}_q)_i + (e_q)_i = k_i - 1, \\ 0 & \text{if } \exists i \ \text{ s.t. } (\overline{e}_q)_i + (e_q)_i > k_i - 1, \end{cases}$$

and entries of $\mathrm{S}_{\Delta_F}(G)$ by equation 1 are

$$s_{\overline{e}_s, e_s} = \begin{cases} (d_{1,k_1}, \ldots, d_{n,k_n})^{\overline{e}_s} & \text{if } \forall\, i \text{ s.t. } (e_s)_i = k_i(\overline{e}_s)_i, \\ 0 & \text{if } \exists\, i \text{ s.t. } k_i(\overline{e}_s)_i < (e_s)_i, \end{cases}$$

for $\overline{e}_s \in \Delta_F$ and e_s in support of G^α for all $\alpha \in \Delta_F$. Therefore

$$(s \cdot q)_{\overline{e}_s \overline{e}_q, e_s + e_q} = \begin{cases} (d_{1,k_1}, \ldots, d_{n,k_n})^{\overline{e}_s + 1} & \text{if } e_s = k_i(\overline{e}_s), \overline{e}_q + e_q = k - 1, \\ 0 & \text{if } \exists\, i \text{ s.t. } k_i(\overline{e}_s)_i < (e_s)_i \\ & \text{or } (\overline{e}_q)_i + (e_q)_i > k_i - 1, \end{cases}$$

i.e., in row $\overline{e}_s \overline{e}_q$ the diagonal element is $(d_{1,k_1}, \ldots, d_{n,k_n})^{\overline{e}_s + 1}$. Since $\overline{e}_s \in \Delta_F$ and $\overline{e}_q \in \overline{\mathcal{Q}}$, where $|\overline{\mathcal{Q}}| = k_1 \cdots k_n$, we have the determinant of A_{R}

$$\det(A_R) = \prod_{\substack{\overline{e}_s \in \Delta_F \\ \overline{e}_q \in \overline{\mathcal{Q}}}} (d_{1,\,k_1}, \ldots, d_{n,k_n})^{\overline{e}_s + 1} = (d_{1,\,k_1}, \ldots, d_{n,k_n})^{(|\Delta_F| + \sum_{\beta \in \Delta_F} \beta)\, k_1 \ldots k_n} .$$

□

By Theorem 1 and the above proposition, we have the following main result.

Theorem 2. *Let F be a polynomial system for which the Cayley-Dixon resultant formulation leads to a square and nonsingular resultant matrix Θ_F whose determinant is* Res(F)*. Then under the multi-univariate composition $F \circ G$,*

$$\mathrm{Res}(F \circ G) = (d_{1,\,k_1}, \ldots, d_{n,\,k_n})^{(\sum_{\alpha \in \overline{\Delta}_F} \alpha + |\Delta_F| + \sum_{\beta \in \Delta_F} \beta)\, k_1 \cdots k_n}\, \mathrm{Res}(F)^{k_1 \cdots k_n} .$$

3.1 Rank Submatrix Construction

In case when the Dixon matrix of the composed polynomials (or any of its factors in Lemma 4) is not square or when the Dixon matrix is rank deficient, one can extract a projection operator from the Dixon matrix by computing the determinant of any maximal minor [8,15]. Since the Dixon matrix $\Theta_{F \circ G}$ can be factored into a product one obtains a similar factorization of a maximal minor,

$$\det_{\max} \left[A_{\mathrm{L}} \times \mathrm{Diag}_{k_1 \cdots k_n}(\Theta_F) \times A_{\mathrm{R}} \right] \quad = \quad \det \left[M_{\mathrm{L}} \times \mathrm{Diag}_{k_1 \cdots k_n}(\Theta_F) \times M_{\mathrm{R}} \right],$$

by selecting appropriate rows M_{L} of A_{L} and columns M_{R} of A_{R}. Furthermore, the well-known Cauchy-Binet formula allows us to expand the determinant of the minor into a sum of products of the form $l \cdot s \cdot r$, where l ranges over determinants of minors of M_{L}, s ranges over determinants of minors of $\mathrm{Diag}_{k_1 \cdots k_n}(\Theta_F)$ and r ranges over determinants of minors of M_{R}.

This leads to a formula similar to the square case (Theorem 2). For details, see the expanded version of this paper [24].

Theorem 3. *For a polynomial system $F = (f_0, f_1, \ldots, f_n)$, composed with univariate polynomials $G = (g_1, \ldots, g_n)$,*

$$\det_{\max}(\Theta_{F\circ G}) = d_{1,k_1}^{\epsilon_1} \cdots d_{n,k_n}^{\epsilon_n} E \left(\gcd \det_{\max}(\Theta_F)\right)^{k_1\cdots k_n},$$

where E is an extraneous factor dependent on the coefficients of G and F.

The above implies that whenever a resultant can be computed by the Cayley-Dixon construction, the resultant also will be decomposable in a similar fashion.

It is an open question what the values of $\epsilon_1, \ldots, \epsilon_n$ are in general and whether the factor E is constant for all selections of maximal minors in $\Theta_{F\circ G}$.

4 Resultant of Composed n-Degree Polynomial System

In this section, we generalize the McKay and Wang formula [19] for the univariate polynomials to n-degree polynomials systems.

Consider the $(m_1, \ldots, m_n)$-degree generic polynomials $f_0, f_1, \ldots, f_n$ where

$$f_j = \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} c_{j,i_1,\ldots,i_n} y_1^{i_1} \cdots y_n^{i_n}, \qquad \text{for} \quad j = 0, 1, \ldots, n,$$

with generic coefficients $c_{j,i_1,\ldots,i_n}$ and variables $y_1, \ldots, y_n$. The composed polynomials $f_i \circ (g_1, \ldots, g_n)$, $i = 0, 1, \ldots, n$, are $(m_1k_1, \ldots, m_nk_n)$-degree as well.

For $\alpha \in \overline{\Delta}_F$, we have $0 \le \alpha_i < (n - i + 1)m_i$, and for $\beta \in \Delta_F$, we have $0 \le \beta_i < im_i$ for $i = 1, \ldots, n$, [21]. Therefore $|\overline{\Delta}_F| = |\Delta_F| = n!\, m_1 \cdots m_n$.

To apply Lemma 4, in the above support, the sum of all points in the support for a particular coordinate $i \in \{1, \ldots, n\}$ is

$$\sum_{\alpha\in\overline{\Delta}_F} \alpha_i = n!\, m_1 \cdots m_n \frac{(n-i+1)m_i - 1}{2}, \qquad \sum_{\beta\in\Delta_F} \beta_i = n!\, m_1 \cdots m_n \frac{im_i - 1}{2}.$$

Substituting into Lemma 4, $\det\left[\mathrm{Diag}_{\overline{Q}}(\Theta_F)\right] = (\det(\Theta_F))^{k_1\cdots k_n}$, and

$$\det[A_L] = \prod_{i=1}^{n} d_{i,\,k_i}^{n!\, m_1\cdots m_n \frac{(n-i+1)m_i-1}{2}\, k_1\cdots k_n},$$

$$\det[A_R] = \prod_{j=1}^{n} d_{j,\,n_j}^{(n!\, m_1\cdots m_n + n!\, m_1\cdots m_n \frac{im_i-1}{2})\ k_1\cdots k_n}.$$

Note that if F and G are generic, then the coefficients of $F \circ G$ will still not have any algebraic relations, and therefore the system $F \circ G$ is generic. By Theorem 2 and the fact that the Dixon matrix is exact for generic n-degree systems [8], we have another main result of the paper.

Theorem 4. *For the unmixed n-degree case,*

$$\mathrm{Res}(F \circ G) = \left(d_{1,k_1}^{m_1} \cdots d_{n,k_n}^{m_n}\right)^{\frac{(n+1)!}{2} m_1\cdots m_2\, k_1\cdots k_n} \mathrm{Res}(F)^{k_1\cdots k_n}.$$

5 Conclusion

This paper studied the behavior of the Cayley-Dixon construction of resultants for multi-univariate composed polynomials. It gave a factorization of the Cayley-Dixon matrix induced by the structure of the composed polynomials and it showed how to efficiently extract the Dixon projection operator utilizing the factorization of the Cayley-Dixon matrix.

In a special case, when polynomials substituted for the variables are $g_i = x_i^k$, the composition problem in the context of Cayley-Dixon construction was analyzed in [18], where it was studied as support scaling. Under this setting, the main result of that paper coincides with Theorem 2. Results presented here are thus a strict generalization.

A new resultant formula has also been derived for multi-univariate composition of n-degree systems.

References

1. Sederberg, T., Goldman, R.: Algebraic geometry for computer-aided design. IEEE Computer Graphics and Applications **6** (1986) 52–59
2. Hoffman, C.: Geometric and Solid modeling. Morgan Kaufmann Publishers, Inc., San Mateo, California 94403 (1989)
3. Morgan, A.: Solving polynomial systems using continuation for Scientific and Engineering problems. Prentice-Hall, Englewood Cliffs, NJ (1987)
4. Chionh, E.: Base points, resultants, and the implicit representation of rational Surfaces. PhD dissertation, Univ. of Waterloo, Dept. of Computer Science (1990)
5. Zhang, M.: Topics in Resultants and Implicitization. PhD thesis, Rice University, Dept. of Computer Science (2000)
6. Bajaj, C., Garrity, T., Warren, J.: On the application of multi-equational resultants. Technical Report CSD-TR-826, Dept. of Computer Science, Purdue (1988)
7. Ponce, J., Kriegman, D.: Elimination Theory and Computer Vision: Recognition and Positioning of Curved 3D Objects from Range. In: Symbolic and Numerical Computation for AI. Academic Press (1992) Donald, Kapur and Mundy (eds.).
8. Kapur, D., Saxena, T., Yang, L.: Algebraic and geometric reasoning using the Dixon resultants. In: ACM ISSAC 94, Oxford, England (1994) 99–107
9. Rubio, R.: Unirational Fields. Theorems, Algorithms and Applications. PhD thesis, University of Cantabria, Santander, Spain (2000)
10. Cheng, C.C., McKay, J.H., Wang, S.S.: A chain rule for multivariable resultants. Proceedings of the American Mathematical Society **123** (1995) 1037–1047
11. Jouanolou, J.P.: Le formalisme du résultant. Adv. Math. **90** (1991) 117–263
12. Minimair, M.: Factoring resultants of linearly combined polynomials. In Sendra, J.R., ed.: Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation, New York, NY, ACM (2003) 207–214 ISSAC 2003, Philadelphia, PA, USA, August 3-6, 2003.
13. Minimair, M.: Computing resultants of partially composed polynomials. In Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V., eds.: Computer Algebra in Scientific Computing. Proceedings of the CASC 2004 (St. Petersburg, Russia). TUM München (2004) 359–366

14. Hong, H., Minimair, M.: Sparse resultant of composed polynomials I. J. Symbolic Computation **33** (2002) 447–465
15. Buse, L., Elkadi, M., Mourrain, B.: Generalized resultants over unirational algebraic varieties. J. Symbolic Computation **29** (2000) 515–526
16. Kapur, D., Saxena, T.: Comparison of various multivariate resultants. In: ACM ISSAC 95, Montreal, Canada (1995)
17. Hong, H.: Subresultants under composition. J. Symb. Comp. **23** (1997) 355–365
18. Kapur, D., Saxena, T.: Extraneous factors in the Dixon resultant formulation. In: ISSAC, Maui, Hawaii, USA (1997) 141–147
19. McKay, J.H., Wang, S.S.: A chain rule for the resultant of two polynomials. Arch. Math. **53** (1989) 347–351
20. Zhang, M., Goldman, R.: Rectangular corner cutting and sylvester $\mathcal{A}$-resultants. In: Proc. of the ISSAC, (St. Andrews, Scotland)
21. Chtcherba, A.D.: A new Sylvester-type Resultant Method based on the Dixon-Bézout Formulation. PhD dissertation, University of New Mexico, Department of Computer Science (2003)
22. Dixon, A.: The eliminant of three quantics in two independent variables. Proc. London Mathematical Society **6** (1908) 468–478
23. Cox, D., Little, J., O'Shea, D.: Using Algebraic Geometry. first edn. Springer-Verlag, New York (1998)
24. Chtcherba, A.D., Kapur, D., Minimair, M.: Cayley-dixon construction of resultants of multi-univariate composed polynomials. Technical Report TR-CS-2005-15, Dept. of Computer Science, University of New Mexico (2005)

A Descartes Algorithm for Polynomials with Bit-Stream Coefficients

Arno Eigenwillig[1], Lutz Kettner[1], Werner Krandick[2], Kurt Mehlhorn[1], Susanne Schmitt[1], and Nicola Wolpert[1]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
{arno, kettner, mehlhorn, sschmitt, wolpert}@mpi-inf.mpg.de
[2] Dept. of Computer Science, Drexel University, Philadelphia, PA, USA
krandick@cs.drexel.edu

Abstract. The Descartes method is an algorithm for isolating the real roots of square-free polynomials with real coefficients. We assume that coefficients are given as (potentially infinite) bit-streams. In other words, coefficients can be approximated to any desired accuracy, but are not known exactly. We show that a variant of the Descartes algorithm can cope with bit-stream coefficients. To isolate the real roots of a square-free real polynomial $q(x) = q_n x^n + \ldots + q_0$ with root separation ρ, coefficients $|q_n| \geq 1$ and $|q_i| \leq 2^\tau$, it needs coefficient approximations to $O(n(\log(1/\rho) + \tau))$ bits after the binary point and has an expected cost of $O(n^4(\log(1/\rho) + \tau)^2)$ bit operations.

1 Introduction

The isolation of the real roots of a real univariate polynomial $q(x) \in \mathbb{R}[x]$ is a fundamental task in computer algebra: given a polynomial q, compute for each of its real roots an interval with rational endpoints containing it and being disjoint from the intervals computed for the other roots. One of the best approaches to root isolation is the Descartes method. It is a bisection method based on the Descartes Rule of Signs to test for roots. Its modern form goes back to Collins and Akritas [1]. It can be formulated to operate on polynomials given in the usual power basis or in the Bernstein basis. For integer coefficients, it typically outperforms other methods. We review it in Section 3. We assume that the coefficients of our polynomials are given as potentially infinite bit-streams, i.e., coefficients are known to arbitrary precision, but, in general, never exactly.

We are the first to make a variant of the Descartes algorithm work in this setting. Our main tools are a sharper analysis of the rule of signs (Lemmas 5 and 6) and randomization (Sections 4.2 and 4.3). Our main result is as follows:

To isolate the real roots of a square-free (= no multiple roots) real polynomial $q(x) = q_n x^n + \ldots + q_0$ with root separation (= minimal distance between any two roots) ρ, coefficients $|q_n| \geq 1$ and $|q_i| \leq 2^\tau$, our algorithm needs coefficient approximations to $O(n(\log(1/\rho) + \tau))$ bits after the binary point and $O(n^4(\log(1/\rho) + \tau)^2)$ bit operations in expectancy.

The cost statement ignores the cost of computing the approximations of the coefficients with the required quality. Observe that the quantities n, ρ and τ are determined by the roots of our polynomial, i.e., the geometry of the problem, and hence the running time of our method is a function of the geometry of the problem.

The restriction to square-free inputs is inherent in the bit-stream setting, since detecting multiple roots is equivalent to testing for zero (cf. $x^2 - a$) and hence impossible.

The paper is structured as follows. In Section 2 we put our work into context and in Section 3 we review the Descartes method. Section 4 is the heart of the paper. We describe and analyze a variant of the Descartes method for polynomials with bit-stream coefficients. In Section 5 we report on some experimental observations.

2 Comparison to Related Work

The Descartes method can be formulated for polynomials in the usual power basis [2,1,3,4] and for polynomials in the Bernstein basis [5–8]. The early work concentrated on polynomials with integer coefficients. More recent work [9,4,8] points out that the Descartes method can be combined with interval arithmetic for increased efficiency and to also handle some, but not all, polynomials with bit-stream coefficients [9, p. 152]. We are the first to exhibit a variant of the Descartes method handling all square-free polynomials with bit-stream coefficients.

Beyond the Descartes method, there is substantial work in numerical analysis on approximating the roots of a real polynomial [10]. Many algorithms were proposed for the simultaneous approximation of all complex roots of a polynomial with bit-stream coefficients. Most algorithms come without a guarantee of convergence. Weyl [11] exhibited the first complete algorithm and Pan [12] surveys the development till about 1995. The currently best algorithm is due to Pan [13]. It applies to polynomials with bit-stream coefficients. Given a polynomial $p(x) = \sum_i p_i x^i = p_n \prod_i (x - z_i)$ of degree n and a precision parameter b, his method computes approximate roots z_i^* such that after suitable renumbering $|z_i^* - z_i| < 2^{2-b/n}$. Here b must be at least $n \log n$ and the computational cost is $O(n \log^2 n(\log^2 n + \log b))$ arithmetic operations (additions and multiplications) on $O(b)$-bit numbers. It is assumed that all roots lie in the unit disk. The algorithm is, in the author's own words, *quite involved, and would require non-trivial implementation work*, and we are not aware of any implementation. Pan's algorithm can be used to isolate real roots. Thus it solves a more general problem (isolation of all roots and not only real roots) and is asymptotically much faster than our algorithm (quadratic dependence on n instead of quartic and linear dependence on $\log(1/\operatorname{sep}(p))$ instead of quadratic). Does this make our contribution obsolete? We believe not: First, because our algorithm is very simple and easily implemented. Second, we expect the algorithm to be superior for small to medium degree polynomials.

3 The Descartes Method in the Bernstein Basis

Fix an integer $n \geq 0$ and boundaries $c < d$ of an interval $[c,d]$. The *Bernstein basis* [14,15] of the vector space $\mathbb{R}[x]_{\leq n}$ of polynomials of degree at most n consists of the *Bernstein polynomials* $B_0^n, B_1^n, \ldots, B_n^n$, where:

$$B_i^n(x) = B_i^n[c,d](x) = \binom{n}{i} \frac{(x-c)^i (d-x)^{n-i}}{(d-c)^n}, \quad 0 \leq i \leq n . \tag{1}$$

If $p(x) = \sum_{i=0}^{n} b_i B_i^n[c,d](x)$, we call $b = (b_n, \dots, b_0)$ the Bernstein representation of p with respect to interval $[c,d]$ and b_0 the *first* and b_n the *last* coefficient. We have $p(c) = b_0$ and $p(d) = b_n$. The Bernstein polynomials form a *non-negative partition of unity*, meaning that $\sum_{i=0}^{n} B_i^n(x) = 1$ and $B_i^n(x) \geq 0$ for all $x \in [c,d]$. This is helpful in bounding error propagation: If $p(x) = \sum_{i=0}^{n} b_i B_i^n(x)$ and $\tilde{p}(x) = \sum_{i=0}^{n} \tilde{b}_i B_i^n(x)$ with $|\tilde{b}_i - b_i| \leq \varepsilon$ for all i, then for all $x \in [c,d]$ it holds that

$$|\sum_i \tilde{b}_i B_i^n(x) - \sum_i b_i B_i^n(x)| \leq \varepsilon \sum_i |B_i^n(x)| = \varepsilon \sum_i B_i^n(x) = \varepsilon\ . \tag{2}$$

The most important property for our purposes is the Descartes Rule of Signs. Let $a = (a_0, \dots, a_n)$ be a finite sequence of real numbers. The *number of sign variations* in a, denoted $\mathrm{var}(a)$, is the number of pairs (i,j) of integers with $0 \leq i < j \leq n$ and $a_i a_j < 0$ and $a_{i+1} = \ldots = a_{j-1} = 0$.

Theorem 1 (The Descartes Rule of Signs). *Let $p(x) = \sum_{i=0}^{n} b_i B_i^n[c,d](x)$ be a polynomial. Then* $\mathrm{var}(b)$ *exceeds the number of zeroes of p in the open interval (c,d) by an even non-negative integer.*

This rule is traditionally stated for the power basis and the interval $(0,\infty)$; see [16] for a proof with historical references. The Bernstein formulation appears in [5–8].

Theorem 1 is the basis for a bisection method for root isolation in exact arithmetic. We start with an interval I guaranteed to contain all real zeroes of p and call *Descartes*(p,I). The procedure *Descartes*(p,I) works as follows: Let $I = (c,d)$, $p(x) = \sum_{i=0}^{n} b_i B_i^n[c,d](x)$ the Bernstein representation of p with respect to the interval (c,d), and $v = \mathrm{var}(b)$. If $v = 0$, return. If $v = 1$, report I as an isolating interval and return. If $v \geq 2$, choose a point $m = \alpha c + (1-\alpha)d$ with $1/4 \leq \alpha \leq 3/4$. (Any $\alpha \in (0,1)$ would work, but a choice near the middle guarantees linear convergence.) If $p(m) = 0$, report the exact root m. Call *Descartes*$(p,(c,m))$ and *Descartes*$(p,(m,d))$.

The Bernstein representations b' and b'' of p with respect to intervals $I' = [c,m]$ and $I'' = [m,d]$ are readily computed from b by *de Casteljau's algorithm* depicted in Figure 1. It operates on a triangular array of numbers. The top row $(b_{0,0}, \dots, b_{0,n})$ is initialized to $b = (b_0, \dots, b_n)$. For $1 \leq j \leq n$, the j-th row $(b_{j,0}, \dots, b_{j,n-j})$ is computed according to

$$b_{j,i} := \alpha b_{j-1,i} + (1-\alpha) b_{j-1,i+1}\ . \tag{3}$$

The result sequences are given by the two sloped sides $b' = (b_{0,0}, b_{1,0}, \dots, b_{n,0})$ and $b'' = (b_{n,0}, b_{n-1,1}, \dots, b_{0,n})$ of the triangle; see, e.g., [14, 3.2/3.3] and [15, Lemma 4.2].

The Descartes method terminates iff the polynomial p is square free. Termination proofs rest on partial converses of Theorem 1 such as the following.

$$\begin{array}{ccccccccccccc}
b_{0,0} & & b_{0,1} & & b_{0,2} & \cdots & b_{0,n-2} & & & b_{0,n-1} & & & b_{0,n} \\
& b_{1,0} & & b_{1,1} & & \cdots & & & b_{1,n-2} & & b_{1,n-1} & & \\
& & b_{2,0} & & b_{2,1} & \cdots & b_{2,n-3} & & & b_{2,n-2} & & & \\
& & & b_{3,0} & & \cdots & & & b_{3,n-3} & & & & \\
& & & & . & \cdots & . & & & & & & \\
& & & & & b_{n,0} & & & & & & &
\end{array}$$

Fig. 1. The de Casteljau triangle in which $b_{j,i} = \alpha b_{j-1,i} + (1-\alpha) b_{j-1,i+1}$

Theorem 2 (Ostrowksi). *Consider a polynomial p and its roots in the complex plane $\mathbb{C}$. Let I be an interval with midpoint m and width $|I|$ and let $v = \mathrm{var}(b)$ be the number of sign variations in the Bernstein representation of p with respect to I.*

If the disc bounded by the circle C centered at m passing through the endpoints of I does not contain any root of p, then $v = 0$ (one-circle theorem).

If the union of the discs bounded by the circles $\underline{C}$ and $\overline{C}$ centered at $m \pm i(\sqrt{3}/6)|I|$ and passing through the endpoints of I contains precisely one simple root of p (which is then necessarily a real root), then $v = 1$ (two-circle theorem).

For a proof see [16]. The circle theorems allow us to bound the depth of the Descartes recursion tree, depending on the distance between the roots. Let p be a non-zero polynomial with roots ξ_1 to ξ_n. We define its *root separation* $\mathrm{sep}(p) = \min\{|\xi_i - \xi_j| \mid i \neq j\}$ as the minimum distance between any two roots; $\mathrm{sep}(p) > 0$ iff p is square-free.

Corollary 3. *The Descartes method applied to any square-free polynomial p and start interval I_0 terminates. The interval at any internal node of the recursion tree has width at least $(\sqrt{3}/2)\,\mathrm{sep}(p)$, the interval at a leaf has width at least $(\sqrt{3}/8)\,\mathrm{sep}(p)$. Given $\sigma \leq \mathrm{sep}(p)$, recursion depth is at most $D(\sigma) := \lfloor \log(|I_0|/\sigma)/\log(4/3) + 3/2 \rfloor$.*

Proof. Consider any interval I for which the Bernstein representation has two or more sign variations. The contrapositive of Theorem 2 tells us: If p has no root in I then there must be a pair of conjugate roots ξ, $\overline{\xi}$ in the disc bounded by C. The diameter of C is $|I|$, hence $|I| \geq |\xi - \overline{\xi}| \geq \mathrm{sep}(p)$. If p has exactly one root ξ' in I, then p has a pair of conjugate roots ξ, $\overline{\xi}$ in the discs bounded by $\overline{C}$ and $\underline{C}$. The diameter of $\overline{C}$ and $\underline{C}$ is $(2/\sqrt{3})|I| \geq |\xi - \xi'| \geq \mathrm{sep}(p)$. If p has two roots ξ, ξ' in I, then $|I| \geq |\xi - \xi'| \geq \mathrm{sep}(p)$. In all three cases, *Descartes*(p, I) generates recursive calls only if $|I| \geq (\sqrt{3}/2)\,\mathrm{sep}(p)$. The interval at a leaf is at least one fourth the length of the interval at its parent. Since the interval length is multiplied by $3/4$ or less in each step, the depth k of an internal node satisfies $|I_0|(3/4)^k \geq (\sqrt{3}/2)\,\mathrm{sep}(p)$ or $k \leq \log(|I_0|/\,\mathrm{sep}(p))/\log(4/3) + 1/2$. □

Proposition 4. *A Descartes recursion tree for a polynomial of degree n has at most n nodes at any depth.*

In the Bernstein basis, this easily seen from the well-known variation diminishing property of repeated linear interpolation. A proof appears in [7, Thm. 10.38].

4 The Descartes Method for Polynomials with Bit-Stream Coefficients

We present an algorithm *Descartes*$_{\mathrm{approx}}$ to isolate the real roots of $p(x) = \sum_{i=0}^{n} b_i B_i^n(x)$ in $(0,1)$. We assume that the coefficients are given as bit-streams; in particular, for any fixed $\varepsilon > 0$, we can compute an approximate coefficient vector $\tilde{b} = (\tilde{b}_0, \ldots, \tilde{b}_n)$ with $|\tilde{b}_i - b_i| \leq \varepsilon$ for all i. We call $\tilde{b}$ an ε-approximate Bernstein representation of p. The pair $(\tilde{b}, \varepsilon)$ specifies an *interval polynomial* $\mathbf{p}(x) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(x)$ such that $\mathbf{p} \supseteq \{p\}$.

We start with a thought experiment. Consider executions of *Descartes* in exact arithmetic both on the exact Bernstein representation b and on its approximation $\tilde{b}$. The only computation *Descartes* ever does with the coefficients is repeated forming of averages as in Eq. (3). The absolute error in the result of such a convex combination is no larger than it is in the inputs. Hence the absolute errors in the de Casteljau triangles in all nodes of the Descartes tree for $\tilde{b}$ are ε-approximations of their counterparts in the tree starting from exact coefficients b. The shape of the exact Descartes tree depends on decisions based on the signs of exact entries. Can we mimic these decisions with intervals?

We call an interval *positive* $(+)$, if it contains only positive numbers; *negative* $(-)$, if it contains only negative numbers; and *indeterminate* (?), if it contains zero. A positive or negative interval is also called *determinate*. For a sequence of coefficient intervals $\mathbf{a} = (\mathbf{a}_0,\ldots,\mathbf{a}_n)$, we define its *set of potential numbers of sign variations* as $\mathrm{var}(\mathbf{a}) = \{\mathrm{var}((a_0,\ldots,a_n)) \mid a_i \in \mathbf{a}_i \text{ for } 0 \le i \le n\}$. For example, we have $\mathrm{var}(([2,3],[-1,1])) = \{0,1\}$, $\mathrm{var}(([2,3],[-1,1],[2,3])) = \{0,2\}$, and $\mathrm{var}(([2,3],[-1,1],[-2,-1])) = \{1\}$. The fact that some $\mathbf{a}_i$ is indeterminate does not imply that $\mathrm{var}(\mathbf{a})$ contains more than one value, as the third example shows.

Consider any node in the approximate Descartes tree. We have an approximate coefficient sequence $\tilde{b}$. Each $\tilde{b}_i$ stands for an interval $\tilde{\mathbf{b}}_i = [\tilde{b}_i - \varepsilon, \tilde{b}_i + \varepsilon]$. Define $\mathrm{var}_\varepsilon(\tilde{b}) = \mathrm{var}(\tilde{\mathbf{b}}_0,\ldots,\tilde{\mathbf{b}}_n)$. Observe that $\mathrm{var}_\varepsilon(\tilde{b})$ contains $\mathrm{var}(b)$. If $\mathrm{var}_\varepsilon(\tilde{b})$ is a singleton or disjoint from $\{0,1\}$, we know what the exact algorithm would do and can do the same.

But what should we do if $\mathrm{var}_\varepsilon(\tilde{b})$ is not a singleton and contains a number less than two? The first solution that comes to mind is to switch to a smaller ε. This will not always solve the problem: Assume we start with a degree-2-polynomial with Bernstein representation $(1,-\beta,\beta)$ with respect to $(0,1)$ where β is any positive irrational number less than one. We split the interval at $1/2$ and obtain $((1-\beta)/4, 0, \beta)$ for the right subinterval. For any approximation of β, the 0 will turn into an interval straddling zero and hence the potential sign variations are $\{0,2\}$. A second solution that comes to mind is to perform recursive calls whenever the set of potential sign variations contains a number larger than one. However, then the procedure might not terminate, namely, when ε is so large that $\mathbf{p}$ contains a non-square-free polynomial. Furthermore: what if the set of potential sign changes contain both zero and one? What if, after subdivision, the last coefficient of b' (first coefficient of b'') is indeterminate, i.e., our polynomial may be zero at the split point?

The last two problems disappear when first and last coefficients are determinate. All problems disappear when first and last coefficients are large. We call $\tilde{b}_i$ *large* if $|\tilde{b}_i| > C\varepsilon$ and *small* otherwise. We will fix the constant $C > 1$ later and prove that if $\tilde{b}_0$ and $\tilde{b}_n$ are large and $\mathrm{var}_\varepsilon(\tilde{b}) \cap \{0,1\} \neq \emptyset$ then $\mathrm{var}_\varepsilon(\tilde{b}'), \mathrm{var}_\varepsilon(\tilde{b}'') \in \{\{0\},\{1\}\}$.

Lemma 5. *Let* $C \ge 4^{n+1}$ *and consider subdivision at* $\alpha \in [1/4, 3/4]$. *If* $\tilde{b}_0$ *and* $\tilde{b}_n$ *are large and positive and* $0 \in \mathrm{var}_\varepsilon(\tilde{b})$ *then all elements of* $\tilde{b}'$ *and* $\tilde{b}''$ *are determinate and positive, i.e.,* $\mathrm{var}_\varepsilon(\tilde{b}') = \mathrm{var}_\varepsilon(\tilde{b}'') = \{0\}$.

Proof. Replace the $\tilde{b}_i$ by modified inputs c_i where $c_i = \tilde{b}_i$ if $\tilde{b}_i$ is determinate and $c_i = 0$ otherwise. This is a change by at most ε. As all entries of the modified de Casteljau triangle $c_{j,i}$ are convex combinations of the inputs, they are all non-negative, and not modified by more than ε. Due to the contribution of c_0 or c_n, resp., any element in the modified output sequences c' and c'' is greater than $4^{-n}C\varepsilon \ge 2\varepsilon$. Thus any element of $\tilde{b}'$ and $\tilde{b}''$ is greater than ε and thus determinate and positive. □

Lemma 6. *Let $C \geq 16^n$ and consider subdivision at $\alpha \in [1/4, 3/4]$. If $\tilde{b}_0$ is large and positive, $\tilde{b}_n$ is large and negative, $\tilde{b}'_n = \tilde{b}''_0$ at the tip of the de Casteljau triangle is large and negative, and $1 \in \mathrm{var}_\varepsilon(\tilde{b})$ then $\mathrm{var}_\varepsilon(\tilde{b}') = \{1\}$ and $\mathrm{var}_\varepsilon(\tilde{b}'') = \{0\}$.*

Proof. As above, we replace all indeterminate elements of $\tilde{b}$ by 0 and denote the elements of the so modified de Casteljau triangle by $c_{j,i}$. The modified input sequence c consists of non-negative followed by non-positive numbers. It is easy to see inductively that all rows of the modified de Casteljau triangle consist of zero or more non-negative elements followed by one or more non-positive elements. Once some row consists entirely of non-positive elements, the same holds for all further rows.

We first prove the claim about c''. The lower tip of the modified triangle is less than $-(C-1)\varepsilon$. A node cannot be less than the minimum of its parents, so there is a path $\mathcal{P}$ of elements less than $-(C-1)\varepsilon$ from row 0 to row n. The elements right of $\mathcal{P}$ are non-positive. Now consider the rightmost element c''_{n-i} in row i of the triangle, for arbitrary i. Go up $0 \leq k \leq i$ times to the left parent until you reach an element of $\mathcal{P}$ in row $i-k$ or end up in row 0 right of the path (with $k = i$). In either case, the last $k+1$ elements of row $i-k$ are non-positive, one of them, say c_*, is less than $-(C-1)\varepsilon$ (namely the path element or c_n), and c''_{n-i} is a convex combination of them. Due to the contribution of c_*, we have $c''_{n-i} < -4^{-k}(C-1)\varepsilon < -2\varepsilon$ and thus $\tilde{b}''_i < -\varepsilon$ holds for all i.

We turn to c'. It begins with $c'_0 > C\varepsilon$ and ends with $c'_n < -(C-1)\varepsilon$. Let

$$i = \min\left\{i \in \{0,\dots,n\}\ ;\ c'_i \leq 0 \text{ or } |c'_i| \leq |c'_{i-1}|/16\right\}\ . \tag{4}$$

Since c'_n is negative, i exists. By minimality of i, we have for all $j < i$ that $c'_j > 0$ and $c'_j > c'_{j-1}/16 > c'_0/16^j > (C/16^{n-1})\varepsilon > 2\varepsilon$. Thus $c'_0, c'_1, \dots, c'_{i-1} > 2\varepsilon$.

Next we will show $c'_{i+1}, \dots, c'_n < -2\varepsilon$. For c'_n, this is already known, so assume $i \leq n-2$. By choice of i, we have $c'_i \leq c'_{i-1}/16$. From $c'_i = \alpha c'_{i-1} + (1-\alpha)c_{i-1,1}$ follows then $c_{i-1,1} = (c'_i - \alpha c'_{i-1})/(1-\alpha) \leq (1-16\alpha)/(16-16\alpha)c'_{i-1}$. This is negative for all $\alpha \in [1/4, 3/4]$, hence $c_{i-1,2} \leq 0$ as well. Now consider

$$\begin{aligned} c'_{i+1} &= \alpha^2 c'_{i-1} + 2\alpha(1-\alpha)c_{i-1,1} + (1-\alpha)^2 c_{i-1,2} \\ &\leq \alpha^2 c'_{i-1} + (\alpha/8)(1-16\alpha)c'_{i-1} = (-\alpha^2 + \alpha/8)c'_{i-1}\ . \end{aligned}$$

The first factor, seen as a function of $\alpha \in [1/4, 3/4]$, takes its maximum $-1/32$ at $\alpha = 1/4$. Hence $c'_{i+1} \leq -(1/32)c'_{i-1} < -(1/32)c'_0/16^{i-1}$. All entries in rows $i+1$ to n are negative and each c'_j from row $i+2$ on receives $\alpha^{j-(i+1)}$ from c'_{i+1}. Thus $c'_j \leq 4^{i-j+1}c'_{i+1} < -4^{i-j+1}32^{-1}16^{1-i}C\varepsilon = -2(C/2^{2i+2j})\varepsilon \leq -2\varepsilon$ for all $j \geq i+1$.

We modified $\tilde{b}$ by at most ε to get c. Hence $\tilde{b}'_0, \dots, \tilde{b}'_{i-1} > \varepsilon$ and $\tilde{b}'_{i+1}, \dots, \tilde{b}'_n < -\varepsilon$, and thus $\mathrm{var}_\varepsilon(\tilde{b}') = \{1\}$. □

Let us now fix $C := 16^n$, satisfying the premises of both lemmas for $\alpha \in [1/4, 3/4]$. Based on these lemmas, we formulate the following exact but yet incomplete procedure *Descartes*$_{\text{approx}}(\tilde{p}, [c,d], \varepsilon)$: Let $\tilde{p}(x) = \sum_{i=0}^n \tilde{b}_i B_i^n[c,d](x)$ be an ε-approximate Bernstein representation of p with respect to the interval $I = [c,d]$. If $\tilde{b}_0$ or $\tilde{b}_n$ is small, abort and signal failure. Otherwise compute $V = \mathrm{var}_\varepsilon(\tilde{b})$, the set of potential values of $\mathrm{var}(b)$. If $V = \{0\}$, return. If $V = \{1\}$, report I as an isolating interval and return. Otherwise, choose a split point $m \in (c + \frac{d-c}{4}, d - \frac{d-c}{4})$ and invoke de Casteljau's algorithm on $\tilde{b}$ to compute approximate Bernstein representations $\tilde{b}'$ and $\tilde{b}''$ for p with respect to intervals $[c,m]$ and $[m,d]$. Call $Descartes_{\mathrm{approx}}(\tilde{p},[c,m],\varepsilon)$ and $Descartes_{\mathrm{approx}}(\tilde{p},[m,d],\varepsilon)$.

Observe that $Descartes_{\mathrm{approx}}$ recurses whenever V contains a value larger than 1. We have shown that whenever $Descartes_{\mathrm{approx}}$ cannot distinguish whether $\mathrm{var}(b)$ is less than two or more than one, this branch of the computation ends in the next recursion step, be it because the tip of the de Casteljau triangle is small or because both new coefficient sequences $\tilde{b}'$ and $\tilde{b}''$ have var_ε equal to $\{0\}$ or $\{1\}$. We conclude that $Descartes_{\mathrm{approx}}$ applied to an ε-approximation of p always terminates (either successfully or by signalling failure) and that the internal nodes of its recursion tree form a subtree of the (exact) Descartes tree. Moreover, if the algorithm terminates successfully, it has determined isolating intervals for the real roots of p.

How can we guarantee that first and last coefficients are large? Key are the observations that first and last coefficients are the values of our polynomial at the interval endpoints, that a polynomial can be small only close to one of its complex roots (see Section 4.1), and that randomization can keep interval endpoints away from the roots (see Section 4.2). We describe two ways of randomization: a local one that selects each split point at random (procedure $Descartes_{\mathrm{rndL}}$) and a global one that selects split points deterministically but runs the entire procedure on a random translate of our input polynomial (procedure $Descartes_{\mathrm{rndG}}$).

4.1 The Smith Bound

We make the link between the complex roots of p and the magnitude of its values through a corollary to the following theorem by Smith [17]. (We state a special case of his result. For its direct proof, see, e.g., [18, Thm. 13].) For a polynomial f, $\mathrm{lcf}(f)$ denotes the absolute value of the leading coefficient (= the coefficient of x^n).

Theorem 7 (Smith bound). *Let g be a polynomial of degree n and let $\xi_1, \ldots, \xi_n$ be pairwise distinct complex numbers. Then for any root z of g there is a ξ_i such that*

$$|z - \xi_i| \le \frac{n\,|g(\xi_i)|}{\mathrm{lcf}(g) \cdot \prod_{j \ne i} |\xi_j - \xi_i|} \,. \tag{5}$$

Corollary 8. *Let f be a square-free polynomial of degree n with complex roots ξ_1 to ξ_n and $\sigma \le \mathrm{sep}(f)$. Let $\tilde{f}(x) = f(x) + e(x)$ be an approximation of f with error term $e(x) = \sum_{i=0}^{n} \varepsilon_i B_i^n[c,d](x)$ where $|\varepsilon_i| \le \varepsilon$ for all i and some fixed $\varepsilon \ge 0$. Let $\gamma \ge 0$ and $z \in [c,d]$. If $|\tilde{f}(z)| \le \gamma$, then there is a root ξ_i of f such that*

$$|z - \xi_i| \le \frac{n(\gamma + \varepsilon)}{\mathrm{lcf}(f) \cdot \prod_{j \ne i} |\xi_j - \xi_i|} \le \frac{n(\gamma + \varepsilon)}{\mathrm{lcf}(f)\ \sigma^{n-1}} \,. \tag{6}$$

Proof. Let $g(x) = f(x) - f(z)$ so that $\mathrm{lcf}(g) = \mathrm{lcf}(f)$ and $g(z) = 0$. By Theorem 7, there is a root ξ_i of f satisfying (5). From (2), we can deduce $|g(\xi_i)| = |f(z)| \le |\tilde{f}(z)| + \varepsilon$. $\square$

4.2 Algorithm *Descartes*$_{\mathrm{rndL}}$

We obtain *Descartes*$_{\mathrm{rndL}}$ from *Descartes*$_{\mathrm{approx}}$ by specifying the choice of split point. In each recursive call, we select the split point m as $m = \alpha c + (1-\alpha)d$ with $\alpha = u/K$, $K = 2^{\lceil 5+\log n \rceil}$, and $u \in \{K/4, K/4+1, \ldots, 3K/4\}$ chosen uniformly at random.

We will show that for at least seven eighth of the possible values of u, m has distance at least $L := L(\varepsilon) := n(C+1)\varepsilon/(\mathrm{lcf}(p)\,\mathrm{sep}(p)^{n-1})$ from every root of p. By the contrapositive of Corollary 8 applied with $\gamma = C\varepsilon$, this guarantees that the approximate value of p at m is greater than $C\varepsilon$ in absolute value. Consider a fixed root ξ of p. Any two adjacent potential split points have distance $(d-c)/K$ and hence there are at most $\lceil 2L/((d-c)/K) \rceil$ values of u for which the distance of m and ξ is less than L. Thus all n roots of p exclude at most $n + 2LKn/(d-c)$ values of u. Since $d - c > \mathrm{sep}(p)/8$ by Corollary 3, this is less than $K/16$ and hence at most one eighth of the possible values for u if $n + 16LKn/\mathrm{sep}(p) \le K/16$ or $16L/\mathrm{sep}(p) \le 1/(16n) - 1/K$. Since $K \ge 32n$, this is fulfilled if $16L/\mathrm{sep}(p) \le 1/(32n)$ or

$$\varepsilon \le \frac{\mathrm{lcf}(p) \cdot \mathrm{sep}(p)^n}{512n^2(C+1)} \,. \tag{7}$$

However, $\mathrm{sep}(p)$ is unknown. Hence we maintain an estimate s for $\mathrm{sep}(p)$. We initialize s a negative power of two, to be specified later, and double $\log(1/s)$, i.e., replace s by s^2, whenever we have indication that s is still too big. For fixed s, we choose ε satisfying (7) by setting $\log(1/\varepsilon) = \lceil n \log(1/s) + 4n + 2\log n + 10 \rceil = O(n \log(1/s))$; this assumes $\mathrm{lcf}(p) \ge 1$. We use two indicators for s being too big: First, we stop when the recursion depth exceeds the bound $D(s)$ from Corollary 3 by more than 1. Second, we call a choice of u and hence m a failure if the last coefficient of the resulting $\tilde{b}'$ (= first coefficient of $\tilde{b}''$) is small. Whenever a choice of u fails, we repeat it. We keep global counters of all choices and failed choices. Whenever the fraction of failed choices is more than half and we have tried at least twelve times, we stop, double $\log(1/s)$ and start over. Once $s \le \mathrm{sep}(p)$, the bound on the recursion depth is no longer a constraint, and the probability of a restart is less than 1/8. To see this, notice that more than half of r random choices failing has probability at most $\binom{r}{\lceil r/2 \rceil}(1/8)^{\lceil r/2 \rceil} \le 2^{-r/2}$, and as we try at least twelve times, the probability of restart is at most $\sum_{r \ge 12}(2^{1/2})^{-r} \le 1/8$.

Initialization. Let $q(x) = \sum_{i=0}^{n} q_i x^i$ be a square-free polynomial in power representation normalized to $q_n \in [1,4)$. (The obvious normalization would be $q_n \in [1,2)$, but with inexact data we need to avoid boundary cases.) We view the coefficients as infinite bit-strings. Let τ be the maximum number of bits before the binary point in any coefficient. All roots of q are bounded by $1 + \max_i |q_i|$ in absolute value (Cauchy bound, [19, Lemma 6.7]) and hence are contained in the open disc of radius $M := 2^{\tau+1}$ about the origin of the complex plane. In particular, the real roots of q are contained in the interval $(-M, +M)$. Let $p(x) = q(4Mx - 2M)/(4M)^n$. Then p has its real roots in $(1/4, 3/4)$ and hence the first and last coefficient of its Bernstein representation with respect to $[0,1]$ are large, $\mathrm{sep}(p) = \mathrm{sep}(q)/2^{\tau+3}$, and $\mathrm{lcf}(p) = \mathrm{lcf}(q)$.

We want to compute an $\varepsilon/2$-approximate Bernstein representation of p with respect to $[0,1]$. (Halving ε is motivated later on.) We compute in fixed-point notation with

$\log(1/\delta)$ bits after the binary point; δ to be determined later. Addition of two such numbers and multiplication with an integer can be done exactly. We start from approximations of the q_i's with error at most δ, compute p in power basis and then convert to Bernstein representation. We have

$$p(x) = \sum_j p_j x^j = q(4Mx - 2M)/(4M)^n = (4M)^{-n} \sum_{0 \le j \le n} x^j \sum_{j \le i \le n} \binom{i}{j} (-2)^{i+j} M^i q_i \,.$$

The factor $\binom{i}{j}$ is an integer less than 2^n and $(4M)^{-n} 2^{i+j} M^i = 2^{i+j+i(\tau+1)-n(\tau+3)}$ is a non-positive power of two (since $i \le n$ and $j \le n$). Hence the p_j's have $O(\tau + n)$ bits before the binary point and error at most $(n+1)2^n\delta$. (To see this, note that the error in each $\binom{i}{j} q_i$ is at most $2^n\delta$; the shifts do not increase the error; each p_j is the sum of at most $n+1$ terms; and the additions do not introduce errors.) We have $p_n = q_n \in [1,4)$. The Bernstein representation of p with respect to $[0,1]$ is given (see [14, 2.8]) by:

$$\sum_{l=0}^{n} b_l B_l^n[0,1](x) = \sum_{l=0}^{n} B_l^n(x) \sum_{k=0}^{l} \frac{\binom{l}{k}}{\binom{n}{k}} p_k = \sum_{l=0}^{n} B_l^n(x) \sum_{k=0}^{l} \frac{l(l-1)\cdots(l-k+1)(n-k)!}{n!} p_k \,.$$

To avoid divisions other than by powers of two, we compute $n!b_l$ instead b_l and later scale as to bring the leading coefficient back to $[1,4)$. We have $l(l-1)\cdots(l-k+1)(n-k)! \le n! \le 2^{n\log n}$. The leading coefficient of $\sum_l n! b_l B_l(x)$ is in $[1,4)n!$. It is brought back into $[1,4)$ by shifting all coefficients to the right by $r \in \log(n!q_n) \pm 1 = O(n\log n)$ bits. The results are the coefficients $\tilde{b}_l$ for use in $\mathit{Descartes}_{\mathrm{rndL}}$. The error in each $\tilde{b}_l$ is bounded by the sum of the errors of the p_j's multiplied by a small power of two accounting for the discrepancy between $1/n!$ and 2^{-r}. Hence the error is at most $(n+1)^2 2^{n+3}\delta$. We want this to be at most $\varepsilon/2$ and therefore choose δ as largest power of two such that $(n+1)^2 2^{n+3}\delta \le \varepsilon/2$. Thus $\log(1/\delta) = \log(1/\varepsilon) + O(n) = O(n\log(1/s))$.

The conversion requires $O(n^2)$ additions and multiplications. The coefficients have at most $O(\tau + n + \log(1/\delta)) = O(\tau + n\log(1/s))$ bits. Multiplications are with numbers of at most $n\log n$ bits and hence the bit complexity of conversion (using high-school multiplication) is $O(n^2 n\log n(\tau + n\log(1/s)))$. This is $O(n^4 \log n \log(1/s))$, since we will choose $s_0 = 2^{-\max(1,\tau/n)}$.

Complexity Analysis. The estimate $\log(1/s)$ runs through the values $2^i\log(1/s_0)$, $i \ge 0$. For each s, we set ε such that $\log(1/\varepsilon) = O(n\log(1/s)) = O(n2^i\log(1/s_0))$. We compute an $\varepsilon/2$-approximate Bernstein representation of p with respect to $(0,1)$ and call $\mathit{Descartes}_{\mathrm{rndL}}(p,(0,1),\varepsilon)$. Its recursion tree has depth at most $D := D(s) + 1 = O(\log(1/s))$ (Corollary 3), and there are at most n nodes on each level (Prop. 4). We perform the arithmetic with $\log(1/\delta_2)$ bits after the binary point; δ_2 fixed as follows. In each node, there are $n(n+1)/2 = O(n^2)$ operations, namely averages, that each introduce an additional error δ_2 but do not add bits before the binary point. The accumulated error in any value is at most $\varepsilon/2 + \frac{n(n+1)}{2} D\delta_2$. With $\frac{n(n+1)}{2} D\delta_2 \le \varepsilon/2$ or $\log(1/\delta_2) = \log(1/\varepsilon) + \log(n(n+1)) + \log(D) = O(n\log(1/s))$, any error is at most ε.

Each averaging operation has bit complexity dominated by the multiplication cost $O(\log n\,(\tau + n + n\log(1/s))) = O(n\log n\log(1/s))$ and hence for any fixed s, the $O(n^3 D)$ operations in total have bit complexity $O(n^4 D\log n\log(1/s)) = O(n^4 \log n \log^2(1/s))$.

The i-th estimate of $\log(1/s)$ is $2^i \log(1/s_0)$, and running *Descartes*$_{\mathrm{rndL}}$ for this value of s costs $O(4^i)\cdot h(n,s_0)$ where $h(n,s) = O(n^4 \log n \log^2(1/s))$. Let $i_1 \geq 0$ be minimal such that $s_1 := 2^{i_1}\log(1/s_0) \geq \log(1/\operatorname{sep}(p))$. The probability that the i-th estimate of s is used is at most $(1/8)^{i-i_1}$ since a call of *Descartes*$_{\mathrm{rndL}}$ fails with probability less than $1/8$ whenever $s \leq \operatorname{sep}(p)$. Hence the expected overall bit complexity is

$$\Big(\sum_{i\leq i_1} 4^i + \sum_{i>i_1} (1/8)^{i-i_1} 4^i\Big)\cdot h(n,s_0) = O(4^{i_1})\cdot h(n,s_0) = h(n,s_1)\ .$$

This means that, asymptotically, the last iteration alone determines the expected cost. Since $\log(1/s_1) = O(\tau/n + \log(1/\operatorname{sep}(p)))$ and $\log(1/\operatorname{sep}(p)) = O(\tau + \log(1/\operatorname{sep}(q))$, we have thus shown

Theorem 9. *Let $q = \sum_{i=0}^n q_i x^i$ with $|q_n| \geq 1$ and $|q_i| \leq 2^\tau$ for all i. The expected bit cost of Descartes$_{\mathrm{rndL}}$ to isolate the real roots of q is is $O(n^4 \log n (\tau + \log(1/\operatorname{sep}(q)))^2)$.*

4.3 Algorithm *Descartes*$_{\mathrm{rndG}}$

Let us now consider another variant of *Descartes* in which we fix $\alpha := 1/2$ globally, meaning that always the interval midpoint is chosen as split point. To keep them away from the roots of p, we replace $p = p_0$ by a random translate $p_\beta(x) = p(x+\beta)$.

We can tighten the recursion depth bound of Corollary 3 to $D(\sigma) := \lfloor \log(1/\sigma) + 2 \rfloor$; and Lemma 6 already holds for $C := 8^n$. (For a proof, replace 8 by 16 in Eq. (4).)

As before, we maintain an estimate s of $\operatorname{sep}(p)$, starting from $s_0 := 2^{-\max(1,\tau/n)}$. The interval $(0,1)$ decomposes into $2^D = 4/s$ intervals of width $s/4$ which we call *elementary intervals*. Any interval I considered by *Descartes*$_{\mathrm{rndG}}$ is a union of elementary intervals (modulo the left endpoint). We call the endpoints of all elementary intervals the *elementary endpoints*. Our goal is to choose β such that any root of p_β has distance greater than L (as defined in Section 4.2) from both endpoints of the elementary interval containing it, so that the approximate value of p_β at all elementary endpoints is greater than $\gamma = C\varepsilon$ in absolute value, so that *Descartes*$_{\mathrm{rndG}}$ is successful. We choose β from

$$\left\{ \frac{u}{K}\cdot\frac{s}{4} \,\middle|\, u \in \{0,1,\ldots,K-1\} \right\} \tag{8}$$

uniformly at random, where the integer K is still to be determined. Consider a fixed root ξ_i of p_0. It excludes at most $1 + (2L/(s/4))K$ values of u. Thus all n roots of p_0 exclude at most $n + (8L/s)nK$ values of u. We want that at least $7/8$ of the values are good and hence require $n + (8L/s)nK \leq K/8$ or $8L/s \leq 1/(8n) - 1/K$. With $K := 2^{\lceil 4+\log n\rceil}$ we obtain the condition $8L(\varepsilon)/s \leq 1/(16n)$, or equivalently,

$$\varepsilon \leq \frac{\operatorname{lcf}(p_0)\cdot s^n}{128n^2(C+1)}\ . \tag{9}$$

We set $\log(1/\varepsilon) = \lceil n\log(1/s) + 3n + 2\log n + 8\rceil = O(n\log(1/s))$ and limit the recursion depth to $D := D(s) + 1$. Whenever *Descartes*$_{\mathrm{approx}}$ aborts with failure, we double $\log(1/s)$ and start again, making a fresh choice for β. Once $s \leq \operatorname{sep}(p)$, every further call to *Descartes*$_{\mathrm{approx}}$ has success probability at least $7/8$.

Initialization. This is very similar to the initialization of *Descartes*$_{\mathrm{rndL}}$. One can compute p_β from p using only addition, multiplication by integers, and bit shifts, without introducing errors. The complexity stays at $O(n^4 \log n \log(1/s))$.

Complexity analysis. Also very similar to *Descartes*$_{\mathrm{rndL}}$. However, the bit complexity of *Descartes*$_{\mathrm{approx}}$ is $O(n^4 \log^2(1/s))$ and hence better by a factor of $\log n$, since $\alpha = u/K$ with numerator length $O(\log n)$ is replaced by $\alpha = 1/2$. This shows

Theorem 10. *Let $q = \sum_{i=0}^{n} q_i x^i$ with $|q_n| \geq 1$ and $|q_i| \leq 2^\tau$ for all i. The expected bit cost of Descartes$_{\mathrm{rndG}}$ to isolate the real roots of q is is $O(n^4(\tau + \log(1/\operatorname{sep}(q)))^2)$.*

5 Some Experiments

We are in the process of conducting experiments[3] on various classes of polynomials with our algorithms *Descartes*$_{\mathrm{rndL}}$ and *Descartes*$_{\mathrm{rndG}}$, implemented in fixed-point arithemtic as detailed in their respective complexity analyses. Some preliminary findings are as follows.

The random choice of the bisection parameter α in *Descartes*$_{\mathrm{rndL}}$ is quite costly in practice when compared to bisection at $\alpha = 1/2$, because the latter does not require multiplication with weights in the averaging step of the de Casteljau algorithm (addition and shift suffice). Hence we suggest to try small denominators of α first, starting with $\alpha = 1/2$, and increasing them in each try up to the original value K. The resulting variant of *Descartes*$_{\mathrm{rndL}}$, called *Descartes*$_{\mathrm{rndL}}^{\mathrm{bias}}$, can be one order of magnitude faster in practice.

Compared to the Bernstein Descartes method implemented with exact integer coefficients and subdivision at $\alpha = 1/2$, *Descartes*$_{\mathrm{rndL}}^{\mathrm{bias}}$ and *Descartes*$_{\mathrm{rndG}}$ tend to be faster for long coefficients.

As expected, experiments also indicate that the running time of our methods, unlike approaches with exact arithmetic, is mostly unaffected by the irrationality of coefficients.

References

1. Collins, G.E., Akritas, A.G.: Polynomial real root isolation using Descartes' rule of signs. In Jenks, R.D., ed.: Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, ACM Press (1976) 272–275
2. Uspensky, J.: Theory of Equations. McGraw-Hill (1948)
3. Krandick, W.: Isolierung reeller Nullstellen von Polynomen. In Herzberger, J., ed.: Wissenschaftliches Rechnen. Akademie-Verlag (1995) 105–154
4. Rouillier, F., Zimmermann, P.: Efficient isolation of a polynomial's real roots. J. Computational and Applied Mathematics **162** (2004) 33–50
5. Lane, J.M., Riesenfeld, R.F.: Bounds on a polynomial. BIT **21** (1981) 112–117
6. Mourrain, B., Vrahatis, M.N., Yakoubsohn, J.C.: On the complexity of isolating real roots and computing with certainty the topological degree. J. Complexity **18** (2002) 612–640
7. Basu, S., Pollack, R., Roy, M.F.: Algorithms in Real Algebraic Geometry. Springer (2003)

[3] See `http://www.mpi-inf.mpg.de/~sschmitt/Descartes`

8. Mourrain, B., Rouillier, F., Roy, M.F.: Bernstein's basis and real root isolation. Rapport de recherche 5149, INRIA-Rocquencourt (2004) `http://www.inria.fr/rrrt/rr-5149.html`.
9. Collins, G.E., Johnson, J.R., Krandick, W.: Interval arithmetic in cylindrical algebraic decomposition. J. Symbolic Computation **34** (2002) 143–155
10. Henrici, P.: Applied and Computational Complex Analysis. Volume 1. Wiley (1974)
11. Weyl, H.: Randbemerkungen zu Hauptproblemen der Mathematik II: Fundamentalsatz der Algebra und Grundlagen der Mathematik. Math. Z. **20** (1924) 131–152
12. Pan, V.: Solving a polynomial equation: Some history and recent progress. SIAM Review **39** (1997) 187–220
13. Pan, V.: Univariate polynomials: Nearly optimal algorithms for numerical factorization and root finding. J. Symbolic Computation **33** (2002) 701–733
14. Prautzsch, H., Boehm, W., Paluszny, M.: Bézier and B-Spline Techniques. Springer (2002)
15. Hoschek, J., Lasser, D.: Fundamentals of computer aided geometric design. A K Peters (1996) Translation of: Grundlagen der geometrischen Datenverarbeitung, Teubner, 1989.
16. Krandick, W., Mehlhorn, K.: New bounds for the Descartes method. Technical report, Drexel University, Dept. of Computer Science (2004) to appear in J. Symbolic Computation, `http://www.cs.drexel.edu/static/reports/DU-CS-04-04.html`.
17. Smith, B.T.: Error bounds for zeros of a polynomial based upon Gerschgorin's theorems. J. ACM **17** (1970) 661–674
18. Bini, D.A., Fiorentino:, G.: Design, analysis, and implementation of a multiprecision polynomial rootfinder. Numerical Algorithms **23** (2000) 127–173
19. Yap, C.K.: Fundamental Problems of Algorithmic Algebra. Oxford University Press (2000)

Real Solving of Bivariate Polynomial Systems

Ioannis Z. Emiris and Elias P. Tsigaridas

Department of Informatics and Telecommunications, National University of Athens, HELLAS
{emiris, et}@di.uoa.gr
http://www.di.uoa.gr/~{emiris,et}

Abstract. We propose exact, complete and efficient methods for 2 problems: First, the real solving of systems of two bivariate rational polynomials of arbitrary degree. This means isolating all common real solutions in rational rectangles and calculating the respective multiplicities. Second, the computation of the sign of bivariate polynomials evaluated at two algebraic numbers of arbitrary degree. Our main motivation comes from nonlinear computational geometry and computer-aided design, where bivariate polynomials lie at the inner loop of many algorithms. The methods employed are based on Sturm-Habicht sequences, univariate resultants and rational univariate representation. We have implemented them very carefully, using advanced object-oriented programming techniques, so as to achieve high practical performance. The algorithms are integrated in the public-domain C++ software library SYNAPS, and their efficiency is illustrated by 9 experiments against existing implementations. Our code is faster in most cases; sometimes it is even faster than numerical approaches.

1 Introduction

Our motivation comes from computer-aided geometric design and computational geometry on curved objects, where predicates rely on the real solving of small algebraic systems and on computing the sign of polynomials evaluated at solutions of such systems. These are crucial in software libraries such as ESOLID ([15]), EXACUS (e.g. [12]), and CGAL (e.g. [8]). Predicates must be decided exactly in all cases, including degeneracies. We focus on bivariate polynomial systems of arbitrary degree. Efficiency is critical because such systems appear in the inner loop of most algorithms, including those for computing the arrangement of algebraic curves or surfaces, the Voronoi diagrams of curved objects, e.g. [6,12] and kinetic data-structures ([11]).

Solving polynomial systems in a real field is an active area of research. There are several algorithms that tackle this problem, cf. e.g. [1,25] and the references therein. Every method designed for the real-solving of algebraic systems could be compared against ours. We focus on those that, in the best of our knowledge, have efficient implementations, enumerated below, and perform experiments against them. Note that we aim at fully accurate computation, in the sense that we avoid any numerical computation and thus we do not compare against software based on homotopies, interval arithmetic or Newton-like methods. Besides our software, only GbRs achieves this goal completely, among the examined implementations. Of course many generalizations of Gröbner bases can lead to solution of this problem, but we chose GbRs and normal forms (solver newmac below) as representatives of them. We also do not test against computer algebra systems, like maple or reduce, or against algorithms for the related but different question on parametric polynomials ([23]).

Our methods are exact, in the sense that they provide rational isolating rectangles for all common roots and calculate their multiplicity. We concentrate on solvers that output this representation. Our methods are also complete, since they can handle all cases, including degeneracies. And they are efficient as testified by their implementation and experiments.

In an earlier paper ([7]), in order to solve *quadratic* bivariate systems, without assuming generic position, we precomputed resultants and static Sturm-Habicht sequences in two variables ([9]) and we combined the rational isolating points with a simple version of rational univariate representation. Here we generalize that approach and use Sturm-Habicht sequences, in a dynamic setting since the polynomial degree is not bounded. Our approach is based on projecting the roots along the two coordinates axes using univariate resultants. We combine the rational isolating points, computed around the resultants' roots, using a specialized version of rational univariate representation, in order to lift them to two dimensions.

Additionally we compute the ordinates of the solutions as algebraic numbers in isolating interval representation, avoiding computations of minimal polynomials. This is important since further computations with the solutions of a system is often required. For example in [13], where computations of Eggers singularities and Milnor numbers are required, or [24], where projections of the roots on three lines are computed, as well as interval refinement in order to compute the critical points of a curve. Our approach allows us to compute the sign of a bivariate polynomial function evaluated over algebraic numbers, all of arbitrary degree. Our approach is similar to [22] and can be easily extended to polynomials with an arbitrary number of variables. However, our approach is more efficient since we use Sturm-Habicht sequences.

The main contribution of this paper is a package for solving bivariate systems and computing the sign of bivariate polynomials evaluated at algebraic numbers, as part of the synaps software library ([5]), which is developed in C++. For this we use modern object-oriented programming techniques, such as partial specialization and traits classes, so as to achieve high performance in practice.

We performed experiments against existing methods and implementations. mapc ([14]) had used Sturm sequences, but did not handle degeneracies. Since mapc is no longer maintained, we do not compare against it. In our tests, we compared against other solvers in synaps [1], namely newmac, which is based on normal forms ([19]), sth, which also uses Sturm-Habicht sequences but uses a double approximation in order to compute the second coordinate of the solutions

[1] www-sop.inria.fr/galaad/logiciels

and is based on the work of [10], and RES, which is based on computing the generalized eigenvalues of a Bézoutian matrix ([2]). Additionally, we test against GbRs [2], through its MAPLE interface, which uses Gröbner bases and rational univariate representation ([21]).

A more recent work is the one of [16], where sparse resultant, rational univariate representation and certified numerical approximations are used so as to solve polynomial systems, with arbitrary number of variables and equations. Their code is not yet freely available. From the running times that they provide it seems that our approach for bivariate systems is faster.

We show that our code compares favourably with other software on most instances. Sometimes it is even faster than methods using some numerical computation; such methods may compromise the accuracy of the output. This shows that for specific instances of real solving, namely when the problem dimension is small, a careful implementation of the Sturm-Habicht approach can be very competitive and even the method of choice.

This paper is organized as follows. The next two sections survey some necessary notions on root multiplicity, and on the theory of Sturm-Habicht sequences. Section 4 presents computations with real algebraic numbers. Section 5 presents our two variants for solving bivariate polynomial system. The following section describes our implementation and experiments. We conclude with open questions.

2 Root Multiplicity

The results of this section can be found for example in [3,1,25]. We follow the approach and the terminology of [13] and [2].

In what follows $\mathbf{D}$ is a ring, $\mathbf{F}$ is a commutative field of characteristic zero and $\overline{\mathbf{F}}$ its algebraic closure. Typically $\mathbf{D} = \mathbb{Z}$, $\mathbf{F} = \mathbb{Q}$ and $\overline{\mathbf{F}} = \overline{\mathbb{Q}}$. Moreover, f, g are bivariate polynomials in $\mathbf{F}[X,Y]$ and $\mathcal{C}_f$ and $\mathcal{C}_g$ are the corresponding affine algebraic plane curves. By $\deg(f)$ we denote the total degree of f, while by $\deg_X(f)$ (respectively $\deg_Y(f)$) denotes the degree of f considered as a univariate polynomial in X (resp. Y) with coefficients in $\mathbf{F}[Y]$ (resp. $\mathbf{F}[X]$).

Let $f, g \in \mathbf{F}[X,Y]$ be two coprime polynomials and $\mathcal{C}_f$, $\mathcal{C}_g$ be their corresponding affine algebraic plane curves, over the field $\overline{\mathbf{F}}$, defined by the equations $(\Sigma) : f(X,Y) = g(X,Y) = 0$. Let $\mathcal{I} =< f, g >$ the ideal that they generate in $\mathbf{F}[X,Y]$ and so the associated quotient ring is $\mathcal{A} = \overline{\mathbf{F}}[X,Y]/\mathcal{I}$. Let the distinct intersection points, which are the distinct roots of (Σ), be $\mathcal{C}_f \cap \mathcal{C}_g = \{\zeta_i = (\alpha_i, \beta_i)\}_{1 \leq i \leq r}$, where $\zeta_i \in \overline{\mathbf{F}}^2$.

The multiplicity of a point ζ_i is

$$\mathrm{mult}(\zeta_i : \mathcal{C}_f \cap \mathcal{C}_g) = \dim_{\overline{\mathbf{F}}} \mathcal{A}_{\zeta_i} < \infty$$

where $\mathcal{A}_{\zeta_i}$ is the local ring obtained by localizing $\mathcal{A}$ at the maximal ideal $\mathcal{I} =< X - \alpha_i, Y - \beta_i >$ ([3]).

[2] http://fgbrs.lip6.fr

If $\mathcal{A}_{\zeta_i}$ is a finite dimensional vector space over $\overline{\mathbf{F}}$, then $\zeta_i = (\alpha_i, \beta_i)$ is an isolated zero of $\mathcal{I}$ and its multiplicity is called the intersection number of the two curves. The finite $\overline{\mathbf{F}}$-algebra $\mathcal{A}$ can be decomposed as a direct sum $\mathcal{A} = \mathcal{A}_{\zeta_1} \oplus \mathcal{A}_{\zeta_2} \oplus \cdots \oplus \mathcal{A}_{\zeta_r}$ and thus $\dim_{\overline{\mathbf{F}}} \mathcal{A} = \sum_{i=1}^{r} \text{mult}(\zeta_i : \mathcal{C}_f \cap \mathcal{C}_g)$.

A polynomial $f \in \mathbf{F}[X,Y]$ or a curve $\mathcal{C}_f$, is called $y-$regular if $\deg(f) = \deg_Y(f)$.

Proposition 1. *Let $f, g \in \mathbf{F}[X,Y]$ be two coprime curves, and let* $\mathrm{p} \in \overline{\mathbf{F}}^2$ *be a point. Then*

$$\text{mult}\,(\mathrm{p} : fg) \geq \text{mult}\,(\mathrm{p} : f)\,\text{mult}\,(\mathrm{p} : g)$$

where equality holds if and only if $\mathcal{C}_f$ and $\mathcal{C}_g$ have no common tangents at p.

Real-solving of (Σ) is equivalent to finding the intersections of $\mathcal{C}_f$ and $\mathcal{C}_g$ in the real plane. We assume that $\mathcal{C}_f$ and $\mathcal{C}_g$ are $y-$regular and that (Σ) is in generic position, meaning that every solution has a distinct $x-$coordinate. This is without loss of generality, since we can achieve this by a linear change of coordinates. A generic linear transformation (shear) of coordinates puts the points of $\mathcal{C}_g \cap \mathcal{C}_g$ in one to one correspondence with roots of the resultant of f and g with respect to Y.

3 Sturm–Habicht Sequences

In this section we present some results for Sturm-Habicht sequences. For more information the reader may refer to [1,10,25].

Let $P, Q \in \mathbb{Z}[X]$ such that $\deg(P) = p$ and $\deg Q = q$ and $P = \sum_{k=0}^{p} a_k X^k, Q = \sum_{k=0}^{q} b_k X^k$. If $i \in \{0, \ldots, \inf(p,q)\}$ we define the polynomial subresultant associated to P and Q of index i, as follows:

$$\mathbf{Sres}_i(P,Q) = \sum_{j=0}^{i} M_i^j x^j$$

where every M_i^j is the determinant of the matrix built with columns $1, 2, \ldots, p+q-2i-1$ and $p+q-i-j$ in the matrix:

$$m_i = \begin{pmatrix} a_p \ldots a_0 & & \\ \ddots & \ddots & \\ & a_p \ldots a_0 & \\ b_p \ldots b_0 & & \\ \ddots & \ddots & \\ & b_p \ldots b_0 & \end{pmatrix}$$

where the coefficients of P and Q are repeated $q-i$ and $p-i$ times respectively. The determinant $M_j^i(P,Q)$ is called the $i-$th principal subresultant coefficient and denoted by $\mathbf{sres}_i$.

Definition 1. *Let P be polynomials in $\mathbb{Z}[X]$ with $p = \deg(P)$. If we write*

$$\delta_k = (-1)^{\frac{k(k+1)}{2}}$$

for every integer k, the Sturm-Habicht sequence associated to P is defined as in the list of polynomials $\{\mathbf{StHa}_j(P)\}_{j=0,\ldots,p}$, where $\mathbf{StHa}_p(P) = P, \mathbf{StHa}_{p-1}(P) = P'$, and for every $j \in \{0, \ldots, p-2\}$:

$$\mathbf{StHa}_j(P) = \delta_{p-j-1}\mathbf{Sres}_j(P, P')$$

For every $j \in \{0, \ldots, p\}$ the principal $j-$th Sturm-Habicht coefficient is defined as:

$$\mathbf{stha}_j(P) = \operatorname*{coeff}_j(\mathbf{StHa}_j(P)),$$

i.e. the coefficient of x^j in the polynomial $\mathbf{StHa}_j(P)$.

It is important to mention that the polynomial $\mathbf{stha}_0$, modulo its sign is the discriminant of P.

Moreover the greatest common divisor of P and P' is obtained as a by-product of the Sturm-Habicht sequence, together with the following equivalence:

$$\mathbf{StHa}_i(P) = \gcd(P, P') \Leftrightarrow \begin{cases} \mathbf{stha}_0(P) = \cdots = \mathbf{stha}_{i-1}(P) = 0 \\ \qquad\qquad \mathbf{stha}_i(P) \neq 0 \end{cases}$$

The Sturm-Habicht sequence has very nice specialization properties. Let P and Q be two polynomials with parametric coefficients, such that their degree does not change after a specialization in the parameters. If we compute their Sturm-Habicht sequence before we specialize the coefficients, the obtained sequence is guaranteed to be valid under every specialization. We use this property so as to compute such a sequence for polynomials in $\mathbb{Z}[X, Y]$, regarding them either as polynomials in $(\mathbb{Z}[X])[Y]$ or in $(\mathbb{Z}[Y])[X]$. The last polynomial in the sequence is the resultant with respect to X or Y, respectively.

Theorem 1. *[1,10] Let f, g square-free and coprime polynomials, such that $\mathcal{C}_f$ and $\mathcal{C}_g$ are in generic position. If*

$$H_j(X, Y) = \mathbf{StHa}_j(f, g) = h_j(X)Y^j + h_{j,j-1}(X)Y^{j-1} + \cdots + h_{j,0}(X)$$

then if $\zeta = (\alpha, \beta) \in \mathcal{C}_f \cap \mathcal{C}_g$ then there exists k, such that

$$h_0(\alpha) = \cdots = h_{k-1}(\alpha) = 0, \quad h_k(\alpha) \neq 0, \quad \beta = -\frac{1}{k}\frac{h_{k,k-1}(\alpha)}{h_k(\alpha)}$$

Note that k is the multiplicity of the solution point.

4 Real Algebraic Numbers and Sign Evaluation

The real algebraic numbers, i.e. those real numbers that satisfy a polynomial equation with integer coefficients, form a real closed field denoted by $\mathbb{R}_{alg} = \overline{\mathbb{Q}}$. From all integer polynomials that have an algebraic number α as root, the one with the minimum degree is called *minimal polynomial.* The minimal polynomial is unique, primitive and irreducible ([25]). In our approach, since we use Sturm-Habicht sequences, it suffices to deal with algebraic numbers, as roots of any square-free polynomial and not as roots of their minimal ones.

In order to represent a real algebraic number we chose the *isolating interval representation.*

Definition 2. *The isolating-interval representation of real algebraic number $\alpha \in \overline{\mathbf{F}}$ is $\alpha \cong (P(X), I)$, where $P(X) \in \mathbf{D}[X]$ is square-free and $P(\alpha) = 0$, $I = [a, b]$, $a, b, \in \mathbf{Q}$ and P has no other root in I.*

Let $\overline{P}$ denote the Sturm-Habicht sequence of P and P'. For a Sturm-Habicht sequence $\overline{P}$, $V_{\overline{P}}(a)$ denotes the number of sign variations of the evaluation of the sequence at a. By $V_{P,Q}(a)$ we denote the sign variations of the Sturm-Habicht sequence of P and Q, evaluated over a.

Theorem 2. *[25] Let $P, Q \in \mathbf{D}[x]$ be relatively prime polynomials and P square-free. If $a < b$ are both non-roots of P and γ ranges over the roots of P in $[a, b]$, then*

$$V_{P,Q}[a, b] := V_{P,Q}(a) - V_{P,Q}(b) = \sum_{\gamma} \operatorname{sign}(P^{'}(\gamma)Q(\gamma)).$$

where $P^{'}$ is the derivative of P.

We can use Sturm-Habicht sequences in order to find the sign of a univariate polynomial, evaluated over a real algebraic number (cf. [9] for degree ≤ 4).

Corollary 1. *Let $Q(X) \in \mathbf{D}[X]$ and a real algebraic number where $\alpha \cong (P, [a, b])$. By th. 2,* $\operatorname{sign}(Q(\alpha)) = \operatorname{sign}(V_{P,Q}[a, b] \cdot Q^{'}(\alpha))$.

Corollary 2. *Th. 2 holds if in place of Q we use $R = \operatorname{prem}(Q, P)$, where* $\operatorname{prem}(Q, P)$, *stands for the pseudo-remainder of Q divided by P.*

Corollary 3. *Using th. 2 we can compare two real algebraic numbers in isolating interval representation.*

Proof. Let two algebraic numbers $\gamma_1 \cong (P_1(x), I_1)$ and $\gamma_2 \cong (P_2(x), I_2)$ where $I_1 = [a_1, b_1]$, $I_2 = [a_2, b_2]$. Let $J = I_1 \cap I_2$. When $J = \emptyset$, or only one of γ_1 and γ_2 belong to J, we can easily order the 2 algebraic numbers. If $\gamma_1, \gamma_2 \in J$, then $\gamma_1 \geq \gamma_2 \Leftrightarrow P_2(\gamma_1) \cdot P_2^{'}(\gamma_2) \geq 0$. We can easily obtain the sign of $P_2^{'}(\gamma_2)$, and from th. 2, we obtain the sign of $P_2(\gamma_1)$. □

The previous tools suffice to compute the sign of a bivariate polynomial function evaluated over two algebraic numbers. Consider $F \in \mathbf{D}[X, Y]$ and $\alpha \cong (A(x), I_1)$ and $\beta \cong (B(X), I_2)$ where $I_1 = [a_1, b_1]$, $I_2 = [a_2, b_2]$. We wish to compute the sign of $F(\alpha, \beta)$.

- Consider F as a univariate polynomial with respect to X and compute the Sturm-Habicht sequence of A and F. Note that the polynomials in the sequence are bivariate.
- Taking advantage of the good specialization properties of Sturm-Habicht sequences, specialize X in the sequence by a_1 and b_1, thus producing two sequences, that contain univariate polynomials.
- For each sequence, for every polynomial in each, compute its sign evaluated at β.
- Finally, count the sign variations for each sequence and the required sign is the difference of these sign variations.

We can extend this approach to polynomials with arbitrary numbers of variables, similar to [22]. However the usage of Sturm-Habicht sequences, instead of generalized Sturm sequences, improves both the theoretical (cf. [1]) and the practical complexity (cf. [4,25]).

5 Two Variants of Bivariate Real Solving

We can make use of th.1, following [10], so as to compute the solution of bivariate polynomial systems. We consider polynomials $f, g \in \mathbb{Q}[X, Y]$, such that $\mathcal{C}_f, \mathcal{C}_g$ are in generic position and we compute the resultant of f, g with respect to Y, which is a polynomial in X. The real solutions of the polynomial correspond to the $x-$coordinates of the solution of the system. Then, using th.1, we lift these solutions in order to determine the $y-$coordinates, as a rational univariate function evaluated over an algebraic number.

Even though the previous approach is straightforward, it has one main disadvantage. The y-coordinates are computed implicitly. If this is all that we want then this is not a problem. However in most cases we want to further manipulate the solutions of the system, i.e. to compare two $y-$coordinates or to count the number of branches of each curve above or below this ordinate. Of course we can always find the minimal polynomial of these algebraic numbers, but this is quite expensive. Thus we chose an alternatively way.

We compute the resultant, using the Sturm-Habicht Sequence, both with respect to Y and X, R_x and R_y respectively. We solve the univariate polynomials R_x and R_y using Sturm sequences (a faster solver like the one in [20], may also be used). Let $\alpha_1 < \cdots < \alpha_k$ and $\beta_1 < \cdots < \beta_l$ be the real roots of R_x and R_y, respectively. For the real roots of R_y we compute rational intermediate points, $q_0 < \beta_1 < q_1 < \cdots < q_{l-1} < \beta_l < q_l$ where $q_j \in \mathbb{Q}, 0 \leq j \leq l$. We can easily compute the intermediate points, since the algebraic numbers are in isolating interval representation.

For every root $\alpha_i, 1 \leq i \leq l$, using th.1, we compute a rational univariate representation of the corresponding y-coordinate, which is without loss of generality, of the form $\gamma_i = \frac{A(\alpha_i)}{B(\alpha_i)}$. Since have already computed the real solutions of R_y, it suffices to determine to which β_j, γ_i equals to, that is to find an index j such that

$$q_j < \frac{A(\alpha_i)}{B(\alpha_i)} < q_{j+1}$$

or, if we assume that $B(\alpha_i) > 0$, this can be checked using cor. 1, then

$$q_j B(\alpha_i) < A(\alpha_i) < q_{j+1} B(\alpha_i)$$

Actually what we really want is to determine the sign of univariate polynomials of the form $U(X) = q_j B(X) - A(X)$ evaluated over the real algebraic numbers that are solutions of $R_x = 0$. This can be done with cor. 1.

However the previous approach works only with the assumption of generic position. This is without loss of generality, since we can apply a transformation of the form $(X, Y) \mapsto (X + aY, Y)$, where a is a random number before the execution of the algorithm, or we can detect non-generic position during the execution ([10,1]), then apply a transformation of the form $(X, Y) \mapsto (X + Y, Y)$ and start the algorithm recursively. However if such a transformation is performed then it is a very hard computational task to apply the inverse transformation so as to represent the solutions to the original coordinate system. Moreover such transformations destroy the sparsity of the system.

In order to overcome such barriers we suggest one more variant for bivariate polynomial system solving. As before we compute the two resultants R_x and R_y and their real solutions α_j and β_j, $1 \leq i \leq k$, $1 \leq j \leq k$, respectively. Then for every pair (a_i, b_j) we test if both f and g vanish. If so, then this pair is the solution. Actually, we do not need to test every pair, since we can take into account the multiplicities of α_i and β_j. The sign of a bivariate polynomial evaluated over two algebraic numbers can be computed using the results of the previous section. This variant is more generic than the previous one, but as one can easily imagine, in most of the cases it is clearly slower. However, it is useful, since a combination of both variants can be used, for example when a non-generic position is detected.

6 Implementation and Experimentation

6.1 Implementation

We have implemented a software package in C++, as part of library SYNAPS [5,18] inside the `namespace ALGEBRAIC`, for dealing with algebraic numbers and bivariate polynomial system solving, which is optimized for small degree. Our implementation is generic in the sense that it can be used with any number type and any polynomial class that supports elementary operations and evaluations and can handle all degenerate cases. We used various advanced C++ programming techniques, such as template specialization, traits classes for number types, etc. Additionally we have precomputed various quantities, and factor several common expressions so as to minimize the computational effort.

In what follows `root_of<RT>` is a class that represents real algebraic numbers, computed as roots of polynomial in isolating interval representation. `UPoly<RT>` is a class for univariate polynomial while `BPoly<RT>` is a class for multivariate polynomials (for our approach we need only the bivariate ones). All classes are parametrized by a ring number type (`RT`).

We present a portion of the functionality that we provide. Actually the presentation is very abstract since various parameters can be determined. For example the univariate solver that can be used and operations between algebraic numbers are not presented here (see [18]). The full description of the functionality, as well as various design and optimization techniques, will be part of a future presentation. The interesting reader may refer to the documentation of SYNAPS for additional details.

- `Seq<root_of<RT> > solve(UPoly<RT>` f`)`
  Solves a univariate polynomial and returns a sorted sequence of real algebraic numbers. For degree up to 4 all the quantities are precomputed using the algorithms of [9]. For higher degrees we use an algorithm for real root isolation based on Sturm-Habicht sequences [4,1,25].
- `int compare(root_of<RT>` α`, root_of<RT>` β`)`
  Compares two algebraic numbers and returns $-1, 0$ or $+1$, depending on the order. For degree up to 4 we use precomputed sequences ([9]). For higher degree we use dynamic (that is computed on the fly) Sturm-Habicht sequences. We use cor. 3 in order to compare them.
- `int sign_at(UPoly<RT>` f`, root_of<RT>` α`)`
  Computes the sign of a univariate polynomial evaluated over an algebraic number returns $-1, 0$ or $+1$. Both the polynomial and the real algebraic number can be of arbitrary degree. We implemented this by using cor. 3.
- `int sign_at(BPoly<RT>` f`, root_of<RT>` γ_x`, root_of<RT>` γ_y`)`
  Computes the sign of a bivariate polynomial evaluated over two real algebraic numbers and returns $-1, 0$ or $+1$. The total degree of the bivariate polynomial, as well as the degree of the algebraic number may be arbitrary. We use cascaded Sturm-Habicht sequences. For total degree of the bivariate polynomial up to 2 and if at least one of the algebraic numbers is of degree less than 5 we use precomputed sequences ([9]).
- `Seq < pair<root_of<RT> > > solve(BPoly<RT>` f_1`, BPoly<RT>` f_2`)`
  Computes the real solutions of a bivariate polynomial system and returns a sequence of pairs of real algebraic numbers sorted lexicographically. If the total degree of the polynomials is less than 3, we use precomputed sequences ([9]). In all the other cases we use the algorithm described in sec. 5.

6.2 Experiments

In order to test our implementation we solved the following systems:

$$
(R_1)\begin{cases} 1+2X-2X^2Y-5XY+X^2+3X^2Y &= 0\\ 2+6X-6X^2Y-11XY+4X^2+5X^3Y &= 0\end{cases}
$$
$$
(R_2)\begin{cases} X^3+3X^2+3X-Y^2+2Y-2=0\\ 2X+Y-3=0\end{cases}
$$
$$
(R_3)\begin{cases} X^3-3X^2-3XY+6X+Y^3-3Y^2+6Y-5=0\\ X+Y-2=0\end{cases}
$$

$$(M_1)\begin{cases} Y^2 - X^2 + X^3 = 0 \\ Y^2 - X^3 + 2X^2 - X = 0 \end{cases}$$
$$(M_2)\begin{cases} X^4 - 2X^2Y + Y^2 + Y^4 - Y^3 = 0 \\ Y - 2X^2 = 0 \end{cases}$$
$$(M_3)\begin{cases} X^6 + 3X^4Y^2 + 3X^2Y^4 + Y^6 - 4X^2Y^2 = 0 \\ Y^2 - X^2 + X^3 = 0 \end{cases}$$
$$(M_4)\begin{cases} X^8 - Y^8 - 1 = 0 \\ X^{10} + Y^{10} - 1 = 0 \end{cases}$$
$$(D_1)\begin{cases} X^4 - Y^4 - 1 = 0 \\ X^5 + Y^5 - 1 = 0 \end{cases}$$
$$(D_1)\begin{cases} -312960 - 2640X^2 - 4800XY - 2880Y^2 + 58080X + 58560Y = 0 \\ -584640 - 20880X^2 + 1740XY + 1740Y + 274920X - 59160Y = 0 \end{cases}$$

where systems $R_{\{1,2,3\}}$ are from [16] and systems $M_{\{1,2,3,4\}}$ are from [2]. The results are on table 1, where times presented are in `msec` and are the average of 100 runs. We performed all tests on a 2.6GHz Pentium with 512MB memory, running Linux, with kernel version 2.6.10. We compiled the programs with `g++`, v. 3.3.5, with option `-O3`.

We test against NEWMAC ([19]). It is a general purpose polynomial system solver. STH, in SYNAPS, is based on Sturm-Habicht sequences and subresultants, following [10]. RES is a bivariate polynomial solver based on the Bézoutian matrix and LAPACK ([2]). For GbRs ([21]) we use its MAPLE interface with 10 digits accuracy, since the source code is not available. S^2 refers to our solver using only the `sign_at` functions, S^2-RUR is our algorithm based the on rational univariate representation.

We have to emphasize that our approach is exact, i.e. it outputs isolating boxes with rational endpoints containing a unique root whose multiplicity is also calculated. This is not the case for STH and RES. STH, uses a double approximation in order to compute the ordinate of the solution. RES works only with doubles, since it has to compute generalized eigenvalues and eigenvectors. These approximations is the reason why they both failed on some tests. NEWMAC also relies on the computation of eigenvalues but in addition computes all the complex solutions of the system.

S^2 is competitive on all data sets, while S^2-RUR is almost always faster than any other solver, even from those that they use `double` arithmetic. However the system of interest is system M_4. Note that this system is very sparse. Sturm-Habicht sequences do not take advantage of the sparsity of a problem. This particular system, is not in generic position, so a linear transformation is applied. Then, at one hand, the sparsity is destroyed and, on the other, the Sturm-Habicht sequences become quite long. We noticed that most of the time is spent for the real solution of the two resultants. This is a strong indication, that a more sophisticated solver for univariate polynomials, like the one in [20], must be adopted.

As for the approach of [16], they quote that, on a faster machine with 3GHz CPU, the timings for solving system R_1, R_2 and R_3 are 2590, 86.5 and 103 msec respectively. So it seems that our approach for bivariate systems is faster. Of course all these are only indications and a more subtle study is required.

Table 1. Experiments on bivariate system solving

msec	R_1	R_2	R_3	M_1	M_2	M_3	M_4	D_1	D_2
S^2	10	1	1	2	3	433	5010	50	1
S^2-RUR	1	1	0.1	1	1	44	1010	11	1
NEWMAC	6	2	3	3	3	20	1020	20	20
RES	0.3	0.3	0.6	0.6	01.2	8.4	150	-	0.5
STH	1	0.2	0.2	0.5	0.4	1.3	-	280	0.4
GbRs	24	22	21	18	23	28	25	25	27

7 Conclusions and Future Work

We are currently working on better bounds on the bit complexity of our algorithms. We expect this investigation to identify possible bottlenecks and lead to better performance in practice. We plan to apply our tools in computing the topology of algebraic curves in 2D and 3D, as well as the topology of surfaces in 3D. Other possible approaches, to be implemented and compared at a practical level, include the adoption of fast Cauchy-index computations ([17]) and Thom's encoding ([1]). Last but not least, we intend to use arithmetic filtering to handle cases that are far from degenerate, so as to improve the speed of our software for generic inputs.

Acknowledgements. The second author thanks Bernard Mourrain, for his continuing help during the implementation of the package. Both authors acknowledge partial support by Kapodistrias, project 70/4/6452 of the Research Council of National University of Athens, by PYTHAGORAS, project 70/3/7392 under the EPEAEK program funded by the Greek Ministry of Educational Affairs and EU, and IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-006413-2 (ACS - Algorithms for Complex Shapes).

References

1. S. Basu, R. Pollack, and M-F.Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2003.
2. L. Busé, B. Mourrain, and H. Khalil. Resultant-based methods for curves intersection problems. Manuscript, 2005.
3. D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry.* Number 185 in Graduate Texts in Mathematics. Springer-Verlag, New York, 1998.
4. J.H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra.* Academic Press, London, 1988.
5. G. Dos Reis, B. Mourrain, R. Rouillier, and P. Trébuchet. An environment for symbolic and numeric computation. In *Proc. Int. Conf. Math. Software*, World Scientific, pages 239–249, 2002.
6. L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Near-optimal parameterization of the intersection of quadrics. In *Proc. ACM SoCG*, pages 246–255, June 2003.
7. I. Z. Emiris and E. P. Tsigaridas. Real algebraic numbers and polynomial systems of small degree. manuscript, 2005. (www.di.uoa.gr/˜et)

8. I.Z. Emiris, A.V. Kakargias, M. Teillaud, E.P. Tsigaridas, and S. Pion. Towards an open curved kernel. In *Proc. ACM SoCG*, pages 438–446, New York, 2004.
9. I.Z. Emiris and E.P. Tsigaridas. Computing with real algebraic numbers of small degree. In *Proc. ESA*, LNCS, pages 652–663. Springer Verlag, 2004.
10. L. Gonzalez-Vega and I. Necula. Efficient topology determination of implicitly defined algebraic plane curves. *Comp. Aided Geom. Design*, 19(9):719–743, 2002.
11. L.J. Guibas, M.I. Karavelas, and D. Russel. A computational framework for handling motion. In *Proc. 6th Work. Algor. Engin. & Experim. (ALENEX)*, 2004.
12. M. Hemmer, E. Schömer, and N. Wolpert. Computing a 3-dimensional cell in an arrangement of quadrics: Exactly and actually! In *Proc. Annual ACM Symp. Comput. Geometry*, pages 264–273, 2001.
13. M. El Kahoui. Computing with algebraic curves in generic position. submitted, 2005.(www.mpi-sb.mpg.de/~elkahoui)
14. J. Keyser, T. Culver, D. Manocha, and S. Krishnan. MAPC: A library for efficient and exact manipulation of algebraic points and curves. In *Proc. Annual ACM Symp. Comput. Geometry*, pages 360–369, New York, N.Y., June 1999. ACM Press.
15. J. Keyser, T. Culver, D. Manocha, and S. Krishnan. ESOLID: A system for exact boundary evaluation. *Comp. Aided Design*, 36(2):175–193, 2004.
16. J. Keyser, K. Ouchi, and M. Rojas. The Exact Rational Univariate Representation for Detecting Degeneracies. In *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. AMS Press, 2004. to appear.
17. T. Lickteig and M.-F. Roy. Semi-algebraic complexity of quotients and sign determination of remainders. *J. Complexity*, 12(4):545–571, December 1996.
18. B. Mourrain, J. P. Pavone, P. Trébuchet, and E. Tsigaridas. SYNAPS, a library for symbolic-numeric computation. In *8th Int. Symp. on Effective Methods in Algebraic Geometry, MEGA*, Italy, May 2005.
19. B. Mourrain and Ph. Trébuchet. Algebraic methods for numerical solving. In *Proc. of the 3rd International Workshop on Symbolic and Numeric Algorithms for Scientific Computing'01 (Timisoara, Romania)*, pages 42–57, 2002.
20. B. Mourrain, M. Vrahatis, and J.C. Yakoubsohn. On the complexity of isolating real roots and computing with certainty the topological degree. *J. Complexity*, 18(2), 2002.
21. F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Journal of Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.
22. T. Sakkalis. Signs of algebraic numbers. *Computers and Mathematics*, pages 131–134, 1989.
23. V. Weispfenning. Solving parametric polynomial equations and inequalities by symbolic algorithms. In Proc. *Computer Algebra in Science and Engineering* World Scientific, 1995
24. N. Wolpert and R. Seidel. On the Exact Computation of the Topology of Real Algebraic Curves. In *Proc. ACM SoCG*, pages 107–115, Pisa, 2004.
25. C.K. Yap. *Fundamental Problems of Algorithmic Algebra.* Oxford University Press, New York, 2000.

Nouvelle Cuisine for the Computation of the Annihilating Ideal of f^s

J. Gago-Vargas[1], M.I. Hartillo-Hermoso[2], and J.M. Ucha-Enríquez[1,⋆]

[1] Depto. de Álgebra, Universidad de Sevilla. Apdo. 1160, E-41080 Sevilla, Spain
gago@algebra.us.es, ucha@us.es
[2] Depto. de Matemáticas, Universidad de Cádiz. Apdo. 40,
E-11510 Puerto Real, Spain
isabel.hartillo@uca.es

Abstract. Let $f_1, \ldots, f_p$ be polynomials in $\mathbf{C}[x_1, \ldots, x_n]$ and let $D = D_n$ be the n-th Weyl algebra. The annihilating ideal of $f^s = f_1^{s_1} \cdots f_p^{s_p}$ in $D[s] = D[s_1, \ldots, s_p]$ is a necessary step for the computation of the Bernstein-Sato ideals of $f_1, \ldots, f_p$.

We point out experimental differences among the efficiency of the available methods to obtain this annihilating ideal and provide some upper bounds for the complexity of its computation.

1 Introduction

Fix two integers $n \geq 1, p \geq 1$ and two sets of variables $(x_1, \ldots, x_n)$ and $(s_1, \ldots, s_p)$. Let us consider $f_1, \ldots, f_p \in \mathbf{C}[x] = \mathbf{C}[x_1, \ldots, x_n]$ and let $D = D_n$ be the n-th Weyl algebra. A polynomial $b(s) \in \mathbf{C}[s] = \mathbf{C}[s_1, \ldots, s_p]$ is said to be a *Bernstein-Sato polynomial associated to* f if the following functional equation holds for a certain $P(s) \in D[s]$:

$$b(s)f^s = P(s)f^{s+\mathbf{1}},$$

where $\mathbf{1} = (1, \ldots, 1)$. These polynomials form an ideal called the *Bernstein-Sato ideal* $\mathcal{B}_f$, or simply $\mathcal{B}$ to abbreviate. Analogous functional equations with respect to vectors different to $\mathbf{1}$ yield other different Bernstein-Sato ideals (see for example [Ba1]).

In [L1] it is proved that $\mathcal{B}$ is not zero. This fact is a generalization of the classical proof of Bernstein ([Be1]) for the case $p = 1$, in which the generator of $\mathcal{B}$ is called *the* Bernstein-Sato polynomial, $b_f(s)$. The analytical work was made in [Bj1] for $p = 1$ and in [Sa1],[Sa2] for $p > 1$.

The roots of $b_f(s)$ encode important algebro-geometrical data (see [Mal1], [H1] or [BS1] to mention only a few) and a complete understanding of all roots for a general f is open. For the case $p > 1$ there is a lot of work to do yet: there are conjectures on the primary decomposition of $\mathcal{B}$, on the conditions over f for $\mathcal{B}$ to be principal, to be zero-dimensional, etc.

In [O1] was presented the first algorithm to find the Bernstein-Sato polynomial, and alternative methods have been proposed to obtain $\mathcal{B}$ in the general case

⋆ All authors partially supported by MTM2004-01165 and FQM-333.

in [OT1], [Ba1] and [BM1]. All these methods have a feature in common: their first step is the computation of the *annihilating ideal* of f^s in $D[s]$, $Ann_{D[s]}f^s$. We recall here some experimental evidences in favor of the method of Briançon-Maisonobe [BM1] with respect to the method of Oaku-Takayama [OT1].

Then we will give upper bounds of the complexity of computing $Ann_{D[s]}f^s$, the previous requirement for both algorithms. To obtain this bounds we use —as far as possible— the techniques and results of [Gr1] on the complexity of solving systems of linear equations over rings of differential operators (that extend the classical polynomial case treated in [Se1]). In particular, we show that the construction of Grigoriev can not be directly generalized to any non-commutative algebra, including the algebra proposed by Briançon-Maisonobe. We prove that the complexity of computing $Ann_{D[s]}f^s$ using the method of [BM1] is that of the calculation of a Gröbner basis in the n-th Weyl algebra with some extra p commutative variables ($2n+p$ variables at most), while in the case of the method [OT1] is the calculation of such a basis in a $(n+p)$-th Weyl algebra with some extra $2p$ variables (so $2n+4p$ variables in all).

We are very grateful to the referees for helping us to clarify our initial version.

2 Preliminaries

In this section we just remind briefly some details of the methods of Briançon-Maisonobe and Oaku-Takayama, respectively.

2.1 Method of Briançon-Maisonobe

In this case the computations are made in the non-commutative algebra

$$R = D_n[s,t] = D_n[s_1,\dots,s_p,t_1,\dots,t_p],$$

an extension of the n-th Weyl algebra D in which the new variables s,t satisfy the relations $[s_i,t_j]=\delta_{ij}t_i$. It is a *Poincaré-Birkhoff-Witt (PBW) algebra*:

Definition 1. *A PBW algebra R over a ring k is an associative algebra generated by finitely many elements $x_1,\dots,x_n$ subject to the relations*

$$Q = \{x_j x_i = q_{ji} x_i x_j + p_{ji}, 1 \le i < j \le n\},$$

where each p_{ji} is a finite k-linear combination of standard *terms $\mathbf{x}^\alpha = x_1^{\alpha_1}\cdots x_n^{\alpha_n}$ and each $q_{ji}\in k$ with the two following conditions:*

1. *There is an* admissible[1] *order $\prec$ on $\mathbf{N}^n$ such that* $\exp(p_{ji}) \prec \exp(x_j x_i)$ *for every $1\le i<j\le n$.*
2. *The standard terms $\mathbf{x}^\alpha$, with $\alpha\in\mathbf{N}^n$, form a k-basis of R as a vector space.*

[1] Here admissible means a total order among the elements of $\mathbf{N}^n$ with $\mathbf{0}$ as least element.

It is possible to compute Gröbner bases in PBW algebras. The book [BGV1] is a good introduction to the subject of effective calculus in this fairly general family.

The following algorithm computes $\mathcal{B}$, starting from

$$I := Ann_R(f^s) = \langle s_j + f_j t_j, \partial_i + \sum_j \frac{\partial f_j}{\partial x_i} t_j, 1 \le i \le n, 1 \le j \le p \rangle.$$

Algorithm 1. You have to:

1. Obtain $J = Ann_{D_n[s]} f^s = \langle G_1 \cap D_n[s] \rangle$ where G_1 is a Gröbner basis of I with respect to any term ordering with variables t_j greater than the others (that is, an *elimination ordering for the* t_j.)
2. $\mathcal{B} = \langle G_2 \cap \mathbf{C}[s] \rangle$, where G_2 is a Gröbner basis of $J + (f_1, \dots, f_p)$ with respect to any term ordering with x_i, ∂_j greater than the s_l.

2.2 Method of Oaku-Takayama

All the computations are made in Weyl algebras. More precisely, starting from

$$I' = \langle t_j - f_j, \sum_{j=1}^{p} \frac{\partial f_j}{\partial x_i} \partial_{t_j} + \partial_i,\ i = 1, \dots, n,\ j = 1, \dots, p \rangle$$

Algorithm 2. You have to:

1. Obtain $J' = I' \bigcap \mathbf{C}[t_1 \partial_{t_1}, \dots, t_n \partial_{t_n}] \langle x, \partial_x \rangle$.
2. $J = Ann_{D_n[s]}(f^s) = J''$, where J'' denotes the ideal generated by the generators of J' after replacing each $t_i \partial_{t_i}$ by $-s_i - 1$.
3. $\mathcal{B} = \langle G_2 \cap \mathbf{C}[s] \rangle$, where G_2 is a Gröbner basis of $J + (f_1, \dots, f_p)$ with respect to any term ordering with x_i, ∂_j greater than the s_l.

The second step above[2] is again the elimination of all the variables but $(s_1, \dots, s_p)$. The computation of

$$I' \cap \mathbf{C}[t_1 \partial_{t_1}, \dots, t_n \partial_{t_n}] \langle x, \partial_x \rangle$$

uses $2n + 4p$ variables, as new variables u_j, v_j for $1 \le j \le p$ are introduced. More precisely, the first calculation is an elimination of these new variables for the ideal

$$\langle t_j - u_j f_j, \sum_{j=1}^{p} \frac{\partial f_j}{\partial x_i} u_j \partial_{t_j} + \partial_i, 1 - u_j v_j, \quad 1 \le i \le n, 1 \le j \le p, \rangle,$$

and some more technical steps must be followed (see [OT1, Procedure 4.1.]).

[2] Often the bottleneck to obtain the Bernstein-Sato ideal is this step. As far as we know, the example for $p = 2$ with $f_1 = x^2 + y^3, f_2 = x^3 + y^2$ is intractable for the available systems.

3 Experimental Data

Here we give some examples for the cases $p = 1, 2$ and $p > 2$ for which it is clear the superiority of Briançon-Maisonobe's method. They have been tested[3] using SINGULAR::PLURAL 2.1 (see [GLS1]) in a PC Pentium IV, 1Gb RAM and 3.06GHz running under Windows XP.

SINGULAR::PLURAL 2.1 is a system for non-commutative general purpose, so the calculations in our algebras are not supposed to be optimal. We present the following data only for the sake of comparing both methods in the same system. In the case of [BM1] method we have used a pure lexicographical ordering, while for [OT1] we have used typical elimination ordering. These are the orderings with best results for each case.

The typical input for SINGULAR::PLURAL 2.1 looks like this for [BM1] method:

```
ring r = 0,(t(1..3),s(1..3),x,y,z,Dx,Dy,Dz),lp;

matrix C[12][12]=0;

C[1,4]=t(1);C[2,5]=t(2);C[3,6]=t(3);C[7,10]=1;C[8,11]=1;C[9,12]=1;

system("PLURAL",1,C);

poly f1 = x*z+y ; poly f2 = x*y+z; poly f3 = y*z+x;

ideal i =s(1)+t(1)*f1,s(2)+t(2)*f2,s(3)+t(3)*f3, Dx +
t(1)*diff(f1,x)+t(2)*diff(f2,x)+t(3)*diff(f3,x), Dy +
t(1)*diff(f1,y)+t(2)*diff(f2,y)+t(3)*diff(f3,y), Dz +
t(1)*diff(f1,z)+t(2)*diff(f2,z)+t(3)*diff(f3,z);

ideal I = std(i);
```

And this one for [OT1] method:

```
ring r = 0,(u,v,x,y,z,t,Dx,Dy,Dz,Dt),(a(1,1),dp);

matrix C[10][10]=0;

C[3,7]=1;C[4,8]=1;C[5,9]=1;C[6,10]=1;

system("PLURAL",1,C);

...
```

[3] The CPU times must be considered as approximations: as it is explained in the SINGULAR::PLURAL 2.1 Manual, the command `timer` is not absolutely reliable due to the shortcomings of the Windows operating system.

1. **Case $p = 1$:** In the following examples $f \in \mathbf{C}[x, y]$ or $f \in \mathbf{C}[x, y, z]$. They have been chosen taking into account Arnold's classification of singularities. E means the memory was exhausted and the system reported an error.

Table 1. CPU times for the computation of $Annf^s$

f	Briançon-Maisonobe's method	Oaku-Takayama's method
$x^3 + xy^2 + z^2$	< 0.01s	0.39s
$x^4 + y^3 + z^2$	< 0.01s	0.39s
$yx^3 + y^3 + z^2$	0.06s	3.97s
$x^3 + y^2 + z^2$	< 0.01s	0.02s
$x^5 + y^2 + z^2$	< 0.01s	4.66s
$x^7 + y^2 + z^2$	< 0.01s	298.56s
$x^4 + y^5 + xy^4$	0.56s	E ($>$ 12h)

2. **Case $p = 2$:** In Table 2 the examples f_1, f_2 are in $\mathbf{C}[x, y]$ or $\mathbf{C}[x, y, z]$.

Table 2. CPU times for the computation of $Annf_1^{s_1} f_2^{s_2}$

f_1	f_2	Briançon-Maisonobe's method	Oaku-Takayama's method
$x^3 + y^2$	$x^2 + y^3$	0.72s	6363.97s
$x^5 + y^3$	$x^3 + y^5$	3.53s	E ($>$ 6h)
$x^7 + y^5$	$x^5 + y^7$	11.84s	E ($>$ 6h)
$x^3 + y^2$	$xz + y$	< 0.01s	9.73s
$x^5 + y^2$	$xz + y$	< 0.01s	1568.59 s
$x^{11} + y^5$	$xz + y$	3s	E ($>$ 6h)

3. **Case $p > 2$:** In Table 3 we have some examples for more than two functions.

Table 3. CPU times for the computation of $Annf_1^{s_1} \cdots f_p^{s_p}$

f_1	f_2	f_3	Briançon-Maisonobe's method	Oaku-Takayama's method
$x + y$	$x - y$	$x^2 + y$	$< 0.01s$	29.46s
$x + y$	$x^2 + y$	$x + y^2$	2.64s	E
$x + y$	$x^2 + y$	$x^2 + y^3$	116.24s	E
$x + y$	$x^2 + y$	$x^3 + y^2$	1728.41s	E

When the single functions $f_1, \dots, f_p$ are "simple" enough (for example, linear) it is possible to obtain $Ann_{D[s_1,\dots,s_p]} f_1^{s_1} \cdots f_p^{s_p}$ for p rather big (say 15 or 20). This ideal can be related to the annihilating ideal of $f = f_1 \cdots f_p$. This idea has been exploited with success in [GHU1] to compute annihilating ideals for f, where f defines very hard examples of arrangements of hyperplanes of theoretical interest. In Table 4 we compare the results of applying this idea in Singular::Plural 2.1 with obtaining directly $Ann_{D[s]} f^s$ using the powerful system Asir (see [N1]).

Table 4. Some arrangements of hyperplanes

$f = f_1 \cdots f_p$	Briançon-Maisonobe's method computing $Ann_{D[s_1,\ldots,s_p]} f_1^{s_1} \cdots f_p^{s_p}$	Asir computing $Ann_{D[s]} f^s$
$xyz(x+y)(x-y)(x-2y-z)$	0.62s	0.93s
$xyz(x-y)(x+y)(x-2y)$	0.05s	0.03s
$xyz(x+y)(x-y)(x+y-z)$	0.06s	3.54s
$xyzu(x+y+z+u)$	0.01s	6.99s
$xyzuv(x+y+z+u+v)$	0.02s	1691.31s
$xyzuvw(x+y+z+u+v+w)$	0.05s	>3 days

4 Complexity

In [Gr1] a bound for the degree of the solutions of a general system of linear equations over the Weyl algebra is given, with a procedure somewhat similar to the one of the commutative case of [Se1]. In this section we study how far the work of Grigoriev is applicable to our PBW algebra R of 2.1. His construction has two different steps: in the first, the given system is reduced to another system in a diagonal form. In the second, it is shown how to normalize the new system in order to eliminate, successively, the variables.

4.1 Diagonalization

We need three technicals lemma to reduce the system to a diagonal form. They generalize analogous lemmas of Grigoriev's paper (see [Gr1, Lemma 1]) and their proofs are, more or less, straightforward. Here deg means the *total degree* of a term, that is, the sum of the exponents of all its variables.

Lemma 1. *Let A be a $(m-1) \times m$ matrix with entries in a Poincaré-Birkhoff-Witt algebra S with a basis of p elements. If* $\deg(a_{ij}) \leq d$, *there exists a nonzero-vector $f = (f_1, \ldots, f_m) \in S^m$ such that $Af = 0$ and* $\deg(f) \leq 2p(m-1)d = N$.

If we work in a noetherian domain (eventually non-commutative), we can always define the *rank* of a finite module as in [St1]. Given a square matrix in a Poincaré-Birkhoff-Witt algebra we say that it is *non-singular* if it has maximal rank, and in this case we can obtain a left quasi-inverse with the precedent lemma:

Lemma 2. *Given a $m \times m$ matrix B over a PBW algebra S as in Lemma 1, non-singular, it has a left quasi-inverse matrix G over S, such that* $\deg(G) \leq N$.

Lemma 3. *Given a system of linear equations over a PBW algebra S, it is defined by a $m \times s$ matrix A of rank r, with its elements* $\deg(a_{ij}) \leq d$ *we can always construct a matrix C, which defines an equivalent system, such that*

$$CA = \begin{pmatrix} C_1 & 0 \\ C_2 & E \end{pmatrix} A = \left(\begin{array}{ccc|c} a_1 & & 0 & \\ & \ddots & & \star \\ 0 & & a_r & \\ \hline & 0 & & 0 \end{array}\right) \tag{1}$$

where E is the identity matrix.

Due to this lemma, we can assume that our system is equivalent to a system in diagonal form:

$$a_k V_k + \sum_{r+1 \le l \le s} a_{k,l} V_l = b_k, \quad 1 \le k \le r, \quad \deg(a_k), \deg(a_{k,l}), \deg(b_k) \le 2pmd.$$

4.2 Normalization

Once the system is in diagonal form, we need to normalize it. To do this, we construct some syzygies, applying Lemma 1 to the submatrix of the first r columns and the column $l > r$. There always exist $h^{(l)}, h_1^{(l)}, \ldots, h_r^{(l)}$ such that:

$$a_k h_k^{(l)} + a_{k,l} h^{(l)} = 0, \quad 1 \le k \le r \quad \deg(h^{(l)}), \deg(h_i^{(l)}) \le 4p^2m^2d$$

The result that gives the normalization in the Weyl algebra is the following one:

Lemma 4 ([Gr1], Lemma 4). *Given $g_1, \ldots, g_t \in D_n$ a family of elements, there is a nonsingular linear transformation of $2n$-dimensional space with basis $x_1, \ldots, x_n, \partial_1, \ldots, \partial_n$ under which:*

$$x_i \to \Gamma_{x_i} = \sum_{j=1}^{n} \gamma_{i,j}^{(1,1)} x_j + \sum_{j=1}^{n} \gamma_{i,j}^{(1,2)} \partial_j;$$

$$\partial_i \to \Gamma_{\partial_i} = \sum_{j=1}^{n} \gamma_{i,j}^{(2,1)} x_j + \sum_{j=1}^{n} \gamma_{i,j}^{(2,2)} \partial_j$$

such that the following relations hold:

$$\Gamma_{x_i}\Gamma_{\partial_i} = \Gamma_{\partial_i}\Gamma_{x_i} - 1; \quad \Gamma_{x_i}\Gamma_{x_j} = \Gamma_{x_j}\Gamma_{x_i}$$

$$\Gamma_{\partial_i}\Gamma_{\partial_j} = \Gamma_{\partial_j}\Gamma_{\partial_i} \quad \Gamma_{\partial_i}\Gamma_{x_j} = \Gamma_{x_j}\Gamma_{\partial_i} \quad i \ne j,$$

and if we denote by Γ_{g_i} the transformed of g_i with the indicated linear transformation, we have $\Gamma_{g_i} = \partial_n^{\deg(g_i)} + \widetilde{\Gamma_{g_i}}$.

Remark 1. The main fact in the proof of the last Lemma 4 is that the matrices of the linear transformations defined by the relations in the Weyl algebra are a transitive group.

Let $\{g_1, \dots, g_t\}$ be a set of elements in $R = \mathbf{C}[s, t, x_1, \dots, x_n, \partial_1, \dots, \partial_n]$. Let us see why we can not assure the existence of a linear transformation Γ that produces

$$\Gamma_{g_i} = v^{\deg(g_i)} + \widetilde{\Gamma_{g_i}},$$

where v is a single variable.

A general linear transformation as the one postulated in Lemma 4 has the form:

$$\begin{array}{rl} s \rightarrow \Gamma_s = & \alpha_1 s + \beta_1 t + \sum_{j=1}^n \gamma_j^{(s,1)} x_j + \sum_{j=1}^n \gamma_j^{(s,2)} \partial_j \\ t \rightarrow \Gamma_t = & \alpha_2 s + \beta_2 t + \sum_{j=1}^n \gamma_j^{(t,1)} x_j + \sum_{j=1}^n \gamma_j^{(t,2)} \partial_j \\ x_i \rightarrow \Gamma_{x_i} = & \alpha_i^{(1)} s + \beta_i^{(1)} t + \sum_{j=1}^n \gamma_{i,j}^{(1,1)} x_j + \sum_{j=1}^n \gamma_{i,j}^{(1,2)} \partial_j \\ \partial_i \rightarrow \Gamma_{\partial_i} = & \alpha_i^{(2)} s + \beta_i^{(2)} t + \sum_{j=1}^n \gamma_{i,j}^{(2,1)} x_j + \sum_{j=1}^n \gamma_{i,j}^{(2,2)} \partial_j \end{array}$$

and it has to verify the following relations:

$$\begin{array}{lll} (1)\ \Gamma_s\Gamma_t = \Gamma_t\Gamma_s + \Gamma_t; & (2)\ \Gamma_s\Gamma_{x_i} = \Gamma_{x_i}\Gamma_s; & (3)\ \Gamma_s\Gamma_{\partial_i} = \Gamma_{\partial_i}\Gamma_s; \\ (4)\ \Gamma_t\Gamma_{x_i} = \Gamma_{x_i}\Gamma_t; & (5)\ \Gamma_t\Gamma_{\partial_i} = \Gamma_{\partial_i}\Gamma_t; & (6)\ \Gamma_{x_i}\Gamma_{\partial_i} = \Gamma_{\partial_i}\Gamma_{x_i} - 1; \\ (7)\ \Gamma_{x_i}\Gamma_{x_j} = \Gamma_{x_j}\Gamma_{x_i}; & (8)\ \Gamma_{\partial_i}\Gamma_{\partial_j} = \Gamma_{\partial_j}\Gamma_{\partial_i}; & (9)\ \Gamma_{x_i}\Gamma_{\partial_j} = \Gamma_{\partial_j}\Gamma_{x_i} \end{array}$$

From relation (1), we obtain $\alpha_2 = \gamma_j^{(t,1)} = \gamma_j^{(t,2)} = 0$ for all j, so $\Gamma_t = \beta_2 t$. The change must be nonsingular, so we have $\beta_2 \neq 0$, and again using relation (1) we deduce that $\alpha_1 = 1$. Using relation (4), we obtain that $\alpha_i^{(1)} = 0$ for all i, and with relation (5) that $\alpha_i^{(2)} = 0$ for all i.

By relation (2) (Γ_s commutes with Γ_{x_i}) we have $\beta_i^{(1)} = 0$, and relation (3) gives $\beta_i^{(2)} = 0$. So Γ must verify:

$$\begin{array}{rl} s \rightarrow \Gamma_s = & s + \beta_1 t + \sum_{j=1}^n \gamma_j^{(s,1)} x_j + \sum_{j=1}^n \gamma_j^{(s,2)} \partial_j \\ t \rightarrow \Gamma_t = & \beta_2 t \\ x_i \rightarrow \Gamma_{x_i} = & \sum_{j=1}^n \gamma_{i,j}^{(1,1)} x_j + \sum_{j=1}^n \gamma_{i,j}^{(1,2)} \partial_j \\ \partial_i \rightarrow \Gamma_{\partial_i} = & \sum_{j=1}^n \gamma_{i,j}^{(2,1)} x_j + \sum_{j=1}^n \gamma_{i,j}^{(2,2)} \partial_j \end{array}.$$

Due to relations from (6) to (9) (between Γ_{x_i} and Γ_{∂_j}) we have that the submatrix

$$\begin{pmatrix} \gamma_{i,j}^{(1,1)} & \gamma_{i,j}^{(1,2)} \\ \gamma_{i,j}^{(2,1)} & \gamma_{i,j}^{(2,2)} \end{pmatrix}$$

verifies the relations of Lemma 4, and in addition, from the relations with Γ_s it verifies

$$\sum \gamma_i^{(s,1)} \gamma_{i,i}^{(1,2)} = \sum \gamma_i^{(s,2)} \gamma_{i,i}^{(1,1)} \quad \sum \gamma_i^{(s,1)} \gamma_{i,i}^{(2,2)} = \sum \gamma_i^{(s,2)} \gamma_{i,i}^{(2,1)}.$$

Anyway if we take for example tx_1, the requirements for Γ produce

$$\Gamma_{tx_1} = \beta_2 t \Gamma_{x_1} \neq v^2 + \widetilde{\Gamma_{tx_1}}.$$

Thus we can not repeat the second step of the process in our PBW algebra in the same way that appears in [Gr1].

Problem 1. Find a general bound for the solutions of a general linear system over any PBW algebra or, at least, give such a bound for R.

We will not treat this general problem: with the aim of obtaining a bound for the complexity of the annihilating ideal of f^s, we will consider only the particular case of one equation of the type that would produce the definition of the ideal I in section 2.1 or I' in section 2.2. In both cases we are interested in the complexity of computing their Gröbner bases (in different rings), and we do it considering the equivalent problem of computing the *syzygies* of the generators of our respective ideals.

Note 1. In the algorithm of [OT1] the calculations are computed in a Weyl algebra of $2n+4p$ variables in all, or more precisely in a commutative polynomial ring with $n+3p, (x,u,v,t)$ commutative variables extended with $n+p, (\partial_x, \partial_t)$ "differential" variables. Let us denote by A this algebra. The complexity of computing the annihilating ideal of f^s is bounded by the complexity of computing a Gröbner basis in A.

Recall that the complexity in the Weyl algebra is given by the following theorem:

Theorem 3 (Th. 6,[Gr1]). *Given a solvable system in the Weyl algebra D_n:*

$$\sum_{1\leq l\leq s} u_{k,l}V_l = w_k, \quad 1\leq k\leq m$$

with $\deg(u_{k,l}), \deg(w_k) \leq d$. *There exists a solution with* $\deg(V_l) < (md)^{2^{O(n)}}$

As we said before in the Briançon-Maisonobe ring R we can not construct a similar algorithm to bound the degree of a solution for a system in general. But in our very special case, our problem is equivalent to computing the solutions of the equation:

$$(s_1+f_1t_1)V_1+\ldots+(s_p+f_pt_p)V_p+$$
$$(\partial_1+\sum_j \frac{\partial f_j}{\partial x_1}t_j)V_{p+1}+\ldots+(\partial_n+\sum_j \frac{\partial f_j}{\partial x_n}t_j)V_{p+n}=0$$

To simplify notation we write the precedent equation as

$$\sum_l Q_lV_l = 0 \qquad (2)$$

Theorem 4. *Given $f=(f_1,\ldots,f_p)$, the complexity of the computation of the annhilating ideal of f^s in the Briançon-Maisonobe algebra $R=D_n[s_1,\ldots,s_p,t_1,\ldots,t_p]$ is bounded by the complexity of the computation of the syzygies of the elements $\partial_i+\sum_j \frac{\partial f_j}{\partial x_i}t_j$ in the Weyl algebra $D_n[t_1,\ldots,t_p]$.*

Proof. We follow the notations of [Gr1] in this proof. We first compute $h_1^{(l)}, h^{(l)}$ for $2 \leq l \leq n+p$ such that:

$$\begin{array}{lll}(s_1+f_1t_1)h_1^{(2)} & +\,(s_2+f_2t_2)h^{(2)} & =0\\ & \vdots & \\ (s_1+f_1t_1)h_1^{(p)} & +\,(s_p+f_pt_p)h^{(p)} & =0\\ (s_1+f_1t_1)h_1^{(p+1)} & +\,(\partial_1+\sum_j \frac{\partial f_j}{\partial x_1}t_j)h^{(p+1)} & =0\\ & \vdots & \\ (s_1+f_1t_1)h_1^{(p+n)} & +\,(\partial_n+\sum_j \frac{\partial f_j}{\partial x_n}t_j)h^{(p+n)} & =0.\end{array}$$

The aim of these $h^{(l)}$ is to reduce any solution $V=(V_1,\dots,V_{p+n})$ of equation (2) to another one without s_1 from which you can recover V. The process will be repeated for $s_2,\dots,s_p$.

It is easy to see that

$$[s_i+f_it_i, s_j+f_jt_j]=0$$

$$[s_i+f_it_i, \partial_j+\sum_l \frac{\partial f_l}{\partial x_j}t_l] = s_i(\sum_l \frac{\partial f_l}{\partial x_j}t_l)+f_it_i\partial_j-\partial_jf_it_i-(\sum_l \frac{\partial f_l}{\partial x_j}t_l)s_i =$$

$$= t_is_i\frac{\partial f_i}{\partial x_j}+t_i\frac{\partial f_i}{\partial x_j}+\sum_{l\neq i}t_ls_i\frac{\partial f_l}{\partial x_j}+t_if_i\partial_j-t_if_i\partial_j-t_i\frac{\partial f_i}{\partial x_j}-\sum_l \frac{\partial f_l}{\partial x_j}t_ls_i=0.$$

Let us define $h^{(l)}=s_1+f_1t_1$ for all $l\geq 2$. We make the division of the V_l of equation (2), $l\geq 2$ by $h^{(l)}$ with respect to a lexicographical ordering with s_1 greater than any other variable. We obtain a remainder $\bar{V}_l$ such that $\deg_{s_1}(\bar{V}_l) < \deg_{s_1}(h^{(l)})=1$, so it has no s_1. So $V_l = h^{(l)}\bar{\bar{V}}_l+\bar{V}_l$, and adding the relation $Q_1h_1^{(l)}+Q_lh^{(l)}=0$ multiplied by $-\bar{\bar{V}}_l$ to equation (2), we obtain:

$$Q_1\bar{V}_1+Q_2\bar{V}_2+\cdots+Q_{n+p}\bar{V}_{n+p}=0$$

with $Q_i, \bar{V}_i$ without s_1 for $i\geq 2$, so $\bar{V}_1=0$, where $\bar{V}_1=V_1-h_1^{(2)}\bar{\bar{V}}_2-\cdots-h_1^{(n+p)}\bar{\bar{V}}_{n+p}$, and we have the new equation:

$$Q_2\bar{V}_2+\cdots+Q_{n+p}\bar{V}_{n+p}=0$$

in a Briançon-Maisonobe algebra $\mathbf{C}[s_2,\dots,s_p,t_1,\dots,t_p,x,\partial]$.

Repeating the process for $Q_2,\dots,Q_p$, we reduce our problem to solving:

$$(\partial_1+\sum_j \frac{\partial f_j}{\partial x_1}t_j)V_{p+1}+\ldots+(\partial_n+\sum_j \frac{\partial f_j}{\partial x_n}t_j)V_{p+n}=0$$

in the Weyl algebra $D_n[t_1,\dots,t_p]$.

As a consequence of 4, the bound for the complexity of computing the annihilating ideal of f^s in R is bounded by the complexity of computing a Gröbner basis in a Weyl algebra with $3p$ *variables less* that the one required by the method of [OT1]. Although the complexity of computing these objects in any case is known to be double exponential (with respect to the number of variables and the total degree of the generators of the ideal) by Theorem 3, it is clear that the reduction of $3p$ variables of [BM1] is a significant advantage in practice as it is shown in the examples (see section 3). The theoretical superiority of the method of [BM1] is an open problem.

Problem 2. Is the bound proposed in this work is reached *a la* Mayr-Meyer ([MM1])? (that is to say, find an example of annihilating ideal with this worst complexity).

References

[Ba1] Bahloul, R. Algorithm for computing Bernstein-Sato ideals associated with a polynomial mapping. Journal of Symbolic Computation **32**, 643–662, 2001.

[Be1] Bernstein, I.N. The analytic continuation of generalized functions with respect to a parameter. Funct. Anal. **6** 273–285, 1972.

[Bj1] Bjork, J.E. Dimensions over Algebras of Differential Operators. Preprint, 1973.

[BM1] Briançon, J. and Maisonobe, Ph. *Remarques sur l'idéal de Bernstein associé à des polynômes.* PUMA, 650 (preprint). 2002.

[BS1] Budur, N. and Saito M. Multiplier ideals, V-filtration and spectrum. arXiv:math.AG/0305118, 2003.

[BGV1] Bueso, J.L. Goméz-Torrecillas, J. and Verschoren, A. *Algorithmic methods in non-commutative algebra. Applications to quantum groups.* Mathematical modelling. Theory and applications **17**, Kluwer Academic Publishers. 2004.

[GHU1] Gago-Vargas, J., Hartillo-Hermoso, M.I. and Ucha-Enríquez, J.M. On the computation of the annihilating ideal of a product of polynomials. In preparation.

[GLS1] G.-M. Greuel, V. Levandovskyy, and H. Schönemann. SINGULAR::PLURAL 2.1. A Computer Algebra System for Noncommutative Polynomial Algebras. Centre for Computer Algebra, University of Kaiserslautern (2003). `http://www.singular.uni-kl.de/plural`.

[Gr1] Grigoriev, D. Complexity of solving systems of linear equations over the rings of differential operators. In *Effective methods in algebraic geometry (Castiglioncello, 1990)*, Progr. Math., **94**. Birkhauser Boston, MA 195–202. 1991.

[H1] Hamm, H.A. Remarks on asymptotic integrals, the polynomial of I.N. Bernstein and the Picard-Lefschetz monodromy. In *Several complex variables (Proc. Sympos. Pure Math. Vol XXX, Part 1, Williams Coll., Williamstown, Mass. 1975)*, 31–35. Amer. Math. Soc., Providence, R.I. 1977.

[K1] Kashiwara, M. B-functions and holonomic systems. Rationality of roots of B-functions. Invent. Math. **38** (1) 33–53. 1976.

[L1] Lichtin, B. Generalized Dirichlet series and b-functions. Compositio Math **65** 81–120. 1988.

[Mal1] Malgrange, B. Le polynôme de Bernstein d'une singularité isolée. In *Fourier integral operators and partial differential equations (Colloq. Internat., Univ. Nice, Nice, 1974)*. Lecture Notes in Math. **459** Springer–Verlag, New York 98–119. 1975.

[MM1] Mayr, E.W. and Meyer, A.R. The complexity of the word problems for commutative semigroups and polynomial ideals. Adv. in Math. **46(3)**, 305–329. 1982.

[N1] M. Noro, T. Shimoyama and Takeshima, T. A Computer Algebra System **Risa/Asir** (2000). Available at `ftp://archives.cs.ehime-u.ac.jp/pub/asir2000/`.

[O1] Oaku, T. An algorithm of computing b-functions. Duke Math. J. **87** 115–132 1997.

[OT1] Oaku, T. Takayama, N. An algorithm for de Rham cohomology groups of the complement of an affine variety via D-module computation. Journal of Pure and Applied Algebra **139** 201–233 1999.

[Sa1] Sabbah, C. Proximité évanescente I. La structure polaire d'un D-Module. Apendice en collaboration avec F.J. Castro Jiménez. Compositio Math. **62**, 283–328. 1987.

[Sa2] Sabbah, C. Proximité évanescente II. Equations fonctionelles pour plusieurs fonctions analytiques. Compositio Math. **64**, 213–241. 1987.

[Se1] Seidenberg, A. Constructions in algebra. Trans. Amer. Math. Soc. **197** 273–313. 1974.

[St1] Stafford, J. T. Module structure of Weyl algebras. J. London Math. Soc. **(2) 18**, no. 3, 429–442. 1978.

Janet-Like Monomial Division

Vladimir P. Gerdt[1] and Yuri A. Blinkov[2]

[1] Laboratory of Information Technologies,
Joint Institute for Nuclear Research, 141980 Dubna, Russia
`gerdt@jinr.ru`
[2] Department of Mathematics and Mechanics,
Saratov University, 410071 Saratov, Russia
`BlinkovUA@info.sgu.ru`

Abstract. In this paper we introduce a new type of monomial division called Janet-like, since its properties are similar to those of Janet division. We show that the former division improves the latter one. This means that a Janet divisor is always a Janet-like divisor but the converse is generally not true. Though Janet-like division is not involutive, it preserves all algorithmic merits of Janet division, including Noetherianity, continuity and constructivity. Due to superiority of Janet-like division over Janet division, the algorithm for constructing Gröbner bases based on the new division is more efficient than its Janet division counterpart.

1 Introduction

In [1] we introduced the concept of involutive division as an underlying notion for theory of involutive Gröbner bases and designed algorithms for their construction. Then, a modified concept of involutive division was introduced in [2] together with another form of involutive algorithms based on this concept.

For a given finite polynomial set and a monomial order, an involutive division partitions the variables into two disjoint subsets called multiplicative and nonmultiplicative. One of such partitions was invented by French mathematician M.Janet in his study [3] of algebraic partial differential equation by means of their transformation (often called completion) to an involutive form [1]. This partition generates Janet division [1] which is one of the most widely used among known involutive divisions. Janet division and related involutive algorithms for completion of polynomial or/and differential systems to involution have been implemented in Reduce, C/C++ [4], Mathematica [5], MuPAD [6], Maple [7], Aldor [8].

One of the main motivations for use of Janet division is its practical computational efficiency. Our present day algorithms [4,9], being optimized versions of those in [1] and implemented for Janet division, demonstrate their superiority over the best implementations [2] of the Buchberger algorithm [10]. Some of the related efficiency aspects are discussed in [9].

[1] This is where the term “involutive” came from.

[2] See our Web page `http://invo.jinr.ru` for experimental comparison with Singular and Magma.

It should be noted, however, that the indicated superiority takes place for most of standard benchmarks, but not for all of them. In addition, there is a class of polynomial systems for which any Janet division algorithm is to be highly inefficient because of a very large Gröbner redundancy of Janet bases. The last means much larger cardinality of Janet bases than that of the reduced Gröbner bases. For those examples the degrees of variables which occur in the leading monomial set of a Gröbner basis form a sparse set in the resulting range of the degrees. It generates a large number of nonmultiplicative prolongations that are involutively head irreducible, and, by this reason, present in the the output Janet basis.

Among such polynomial systems are those generating toric ideals as we already demonstrated in [11]. By the maximality arguments [2], one can expect that any other involutive division will also be inefficient for those systems.

In the given paper we introduce another monomial division which also restricts the conventional division (cf. [9]) and is very similar to Janet division. By this reason we call it Janet-like. The new division, however, is not involutive. Nevertheless, it possesses all merits of Janet division. Moreover, it improves the last division by optimizing the number of operations needed to construct the output Gröbner bases called Janet-like. Janet-like bases and their algorithmic construction are considered in a separate paper [12].

2 Basics

Let $\mathbb{R} = \mathbb{K}[\mathbb{X}]$ be the polynomial ring over the field $\mathbb{K}$ in the indeterminates $\mathbb{X} = \{x_1, \ldots, x_n\}$. By $\mathbb{M}$ we denote the monoid of monomials in $\{x_1^{i_1} \cdots x_n^{i_n} \mid i_k \in \mathbb{N}_{\geq 0},\ 1 \leq k \leq n\}$ in $\mathbb{R}$. By $\deg_i(u)$ we denote the degree of x_i in $u \in \mathbb{M}$ and by $\deg(u) = \sum_{i=1}^{n} \deg_i(u)$ the total degree of u. An admissible monomial order such that

$$x_1 \succ x_2 \succ \cdots \succ x_n \tag{1}$$

will be denoted by $\succ$.

As usual, the conventional divisibility of monomial v by monomial u will be denoted by $u \mid v$. If $u \mid v$ and $\deg(u) < \deg(v)$, i.e. u is a proper divisor of v, we shall write $u \sqsubset v$.

For a polynomial $f \in \mathbb{R} \setminus \{0\}$ its leading monomial, leading term and leading coefficient will be denoted by $\mathrm{lm}(f)$, $\mathrm{lt}(f)$ and $\mathrm{lc}(f)$, respectively. Given a polynomial set $F \subset \mathbb{R} \setminus \{0\}$ and an order $\succ$, $\mathrm{lm}(F)$ will denote the leading monomial set of F. For the ideal in $\mathcal{I} \in \mathbb{R}$ generated by a polynomial set $F \subset \mathbb{R}$ we shall write $\mathcal{I} = \mathrm{Id}(F)$.

Definition 1. *Gröbner basis* [10]. Given an order $\succ$, a finite subset $G \subset \mathbb{R}$ is called a *Gröbner basis* of ideal $\mathcal{I} = \mathrm{Id}(G) \in \mathbb{R}$ if

$$\forall f \in \mathcal{I},\ \exists g \in G \ :\ \mathrm{lm}(g) \mid \mathrm{lm}(f)\,. \tag{2}$$

The basic idea behind the involutive division approach [1,2] is to replace the conventional division in (2) by its certain restriction called involutive. In the present paper, we need, however, more general concepts defined as follows.

Definition 2. (*Restricted division* [9]). A *restricted division* r on $\mathbb{M}$ is a reflexive transitive relation, denoted by $u \mid_r v$ $(u, v \in \mathbb{M})$, such that $u \mid_r v \Longrightarrow u \mid v$. If $u \mid_r v$, then u is *$r-$divisor* of v and v is *r-multiple* of u, respectively.

Definition 3. (*r-basis* [9]). Given an order $\succ$ and a restricted division r, a finite subset $G \subset \mathbb{R}$ is called *$r-$basis* of ideal $\mathcal{I} = \mathrm{Id}(G) \in \mathbb{R}$ if

$$\forall f \in \mathcal{I},\ \exists g \in G\ :\ \mathrm{lm}(g) \mid_r \mathrm{lm}(f)\,.$$

Note, that the whole class of restricted divisions includes the conventional division as well. From Definition 2 it follows that a $r-$basis, if exists, is always a Gröbner basis. It is also easy to reformulate the concept of Gröbner reduction and normal form [10] in terms of the restricted division [9].

A natural way to introduce a restricted monomial division r is to indicate a certain subset $X(u) \subseteq \mathbb{X}$ of indeterminates for a monomial $u \in \mathbb{M}$ and to define for $v \in \mathbb{M}$

$$u \mid_r v \Longleftrightarrow v = u \cdot w,$$

where w belongs to the monoid of power products constructed from the indeterminates in $X(u)$. Besides, Definitions 1 and 3 deal with r-divisors taken from a certain finite monomial set. By this reason, for algorithmic purposes of constructing Gröbner bases, it suffices to define an r-division for an arbitrary finite set of possible monomial divisors.

Involutive divisions [1,2] form an algorithmically interesting class of this sort of restricted divisions. Our concept of involutive division is given by the folowing definition. [3]

Definition 4. (*Involutive division* [1]). We say that an *involutive division* $\mathcal{L}$ is defined on $\mathbb{M}$ if for any nonempty finite monomial set $U \subset \mathbb{M}$ and for any $u \in U$ there defined a subset $M_{\mathcal{L}}(u, U) \subseteq \mathbb{X}$ (possibly empty [4]) of indeterminates whose power products generate submonoid $\mathcal{L}(u, U)$ of $\mathbb{M}$ such that the following holds

1. $v \in U\ \wedge\ u\mathcal{L}(u, U) \cap v\mathcal{L}(v, U) \neq \emptyset \Longrightarrow u \in v\mathcal{L}(v, U)\ \vee\ v \in u\mathcal{L}(u, U)$.
2. $v \in U\ \wedge\ v \in u\mathcal{L}(u, U) \Longrightarrow \mathcal{L}(v, U) \subseteq \mathcal{L}(u, U)$.
3. $u \in V \wedge\ V \subseteq U \Longrightarrow \mathcal{L}(u, U) \subseteq \mathcal{L}(u, V)$.

Indeterminates in $M_{\mathcal{L}}(u, U)$ are called *$\mathcal{L}-$ multiplicative* for u and the remaining indeterminates in $NM_{\mathcal{L}}(u, U) := \mathbb{X} \setminus M_{\mathcal{L}}(u, U)$ are called *$\mathcal{L}-$nonmultiplicative* for u, respectively. If $w \in u\mathcal{L}(u, U)$, then u is called *$\mathcal{L}-$(involutive) divisor* of w.

[3] Another concept [2] also satisfies conditions 1 and 2 for any given monomial set but not necessarily condition 3.

[4] If $M_{\mathcal{L}}(u, U) = \emptyset$, then $\mathcal{L}(u, U) = \{1\}$.

A typical and computationally good [9] representative of involutive division is Janet division.

Definition 5. *Janet division* [1]. Let $U \subset \mathbb{M}$ be a finite set. For each $0 \le i \le n$ partition U into groups labeled by non-negative integers $d_0, \ldots, d_i$ [5]

$$[d_0, d_1, \ldots, d_i] := \{u \in U \mid d_0 = 0, d_1 = \deg_1(u), \cdots, d_i = \deg_i(u)\}. \qquad (3)$$

Indeterminate x_i is *J(anet)-multiplicative* for $u \in U$ if $u \in [d_0, \ldots, d_{i-1}]$ and $\deg_i(u) = \max\{\deg_i(v) \mid v \in [d_0, \ldots, d_{i-1}]\}$.

Below, in accordance to the notations used in Definition 4, we shall write $M_J(u, U)$ and $NM_J(u, U)$ for the sets of J-multiplicative and J-nonmultiplicative indeterminates for monomial $u \in U$, and write $u \mid_J w$ if u is a $J-$divisor of w.

3 Janet-Like Division

In this section we introduce a non-involutive restricted division which improves algorithmic properties of Janet division. In the following, unless mentioned, the monomial subsets of $\mathbb{M}$ are assumed to be finite and nonempty, and polynomial subsets of $\mathbb{R}$ are also assumed to be finite and without zero polynomials.

Definition 6. (*Nonmultiplicative power*). Let $U \subset \mathbb{M}$ be a monomial set and its elements be partitioned into groups as in (3). For every $u \in U$ and $1 \le i \le n$ consider $h_i(u, U) \in \mathbb{N}_{\ge 0}$ given by

$$h_i(u, U) := \max\{\deg_i(v) \mid u, v \in [d_0, \ldots, d_{i-1}]\} - \deg_i(u).$$

If $h_i(u, U) > 0$, then the power $x_i^{k_i}$ where

$$k_i := \min\{\deg_i(v) - \deg_i(u) \mid v, u \in [d_0, \ldots, d_{i-1}],\, \deg_i(v) > \deg_i(u)\}$$

will be called a *nonmultiplicative power* for u.

We shall denote by $NMP(u, U)$ the set of all nonmultiplicative powers for $u \in U$.

Definition 7. (*Janet-like division*). Given a set $U \subset \mathbb{M}$ and $u \in U$, elements of the monoid ideal

$$\mathcal{NM}(u, U) := \{v \in \mathbb{M} \mid \exists w \in NMP(u, U) : w \mid v\} \qquad (4)$$

will be called *$\mathcal{J}$-nonmultipliers* for $u \in U$. Elements in $\mathcal{M}(u, U) := \mathbb{M} \backslash \mathcal{NM}(u, U)$ will be correspondingly called *$\mathcal{J}$-multipliers* for u. Element $u \in U$ will be called a *Janet-like divisor or $\mathcal{J}-$divisor* of $w \in \mathbb{M}$ (denotation $u \mid_{\mathcal{J}} w$) if $w = u \cdot v$ with $v \in \mathcal{M}(u, U)$.

Remark 1. From comparison of Definitions 5 and 7 it follows immediately that every nonmultiplicative power is nothing else then the power of $J-$nonmultiplicative indeterminate. The following example illustrates this obvious fact.

Example 1. $U = \{x_1^5, x_1^2x_2^2x_3, x_1^2x_3^2, x_2^4x_3, x_2x_3^2, x_3^5\} \subset \mathbb{K}[x_1, x_2, x_3]$.

[5] Note that $U = [0]$.

Table 1. Comparison of Janet and Janet-like divisions

Element in U	Division		
	Janet		Janet-like
	M_J	NM_J	NMP
x_1^5	x_1, x_2, x_3	$-$	$-$
$x_1^2x_2^2x_3$	x_2, x_3	x_1	x_1^3
$x_1^2x_3^2$	x_3	x_1, x_2	x_1^3, x_2^2
$x_2^4x_3$	x_2, x_3	x_1	x_1^2
$x_2x_3^2$	x_3	x_1, x_2	x_1^2, x_2^3
x_3^5	x_3	x_1, x_2	x_1^2, x_2

Proposition 1. *Let $U \subset \mathbb{M}$ be a set, $u \in U$ be its element and $w \in \mathbb{M}$ be a monomial. Then Janet-divisibility of w by u implies its Janet-like divisibility, i.e.*

$$u \mid_J w \Longrightarrow u \mid_{\mathcal{J}} w\,.$$

The converse is generally not true.

Proof. By Definition 5, w/u is a power product of $J-$multiplicative variables for u. Since any element in $NMP(u,U)$ is a power of a $J-$nonmultiplicative variable, w is $\mathcal{J}-$multiplier. On the other side, if $x_i^{k_i} \in NMP(u,U)$ and $k_i > 1$, then, in accordance with Definition 7, $u \mid_{\mathcal{J}} u \cdot x_i^{k_i-1}$ whereas $x_i \in NM_J(u,U)$. $\Box$

In the rest of this paper we show that Janet-like division, whereas providing a wider divisibility then Janet division, as the Proposition 1 states, possesses all algorithmic merits of the last division. First, we show that as well as a Janet divisor, a Janet-like divisor is unique.

Proposition 2. *A monomial w cannot have two distinct $\mathcal{J}-$divisors in any monomial set.*

Proof. Suppose there are two $\mathcal{J}-$divisors $u, v \in U$ and $u \neq v$. Let i be the lowest index such that $\deg_i(u) \neq \deg_i(v)$. Without the loss of generality assume that $\deg_i(u) < \deg_i(v)$. Then $u, v \in [d_0, \ldots, d_{i-1}]$ and, in accordance with Definition 6, $x_i^{k_i} \in NMP(u,U)$ where $0 < k_i \leq \deg_i(v) - \deg_i(u)$. Since $v \mid w$, w/u is a multiple of $x_i^{k_i}$, and u cannot $\mathcal{J}-$divide w, a contradiction. $\Box$

Definition 8. (*Completeness*). A monomial set U is called *$\mathcal{J}-$complete* if the equality

$$C_{\mathcal{J}}(U) = C(U) \tag{5}$$

holds, where

$$C_{\mathcal{J}}(U) := \{u \cdot v \mid u \in U, v \in \mathcal{M}(u,U)\}\,, \tag{6}$$

$$C(U) := \{u \cdot w \mid u \in U, w \in \mathbb{M}\}\,. \tag{7}$$

Equality (5) means that any element in the monoid ideal (7) generated by elements in the complete set U has a $\mathcal{J}-$divisor in U. If U contains only distinct monomials, then, by Proposition 2, this ideal is partitioned into the disjoint subsets generated by $\mathcal{J}-$multiples of elements in U.

Lemma 1. *Let $u, v \in U$ and $u \mid_{\mathcal{J}} v \cdot p$, $p \in NMP(u, U)$. Then $u \succ_{lex} v$ where $\succ_{lex}$ is the lexicographical monomial order induced by (1).*

Proof. Assume that $p = x_i^{k_i}$ $(1 \leq i \leq n, k_i > 0)$. If $i = 1$, then $\deg_1(u) > \deg_1(v)$. Indeed, if $d_1 := \deg_1(u) = \deg_1(v)$, then $u, v \in [d_0, d_1]$. Since $p \in NMP(v, U)$, by Definition 6, $p \in NMP(u, U)$ what contradicts $\mathcal{J}$-divisibility $v \cdot p$ by u. If $\deg_1(u) < \deg_1(v)$, then again, by Definitions 6 and 7, $x_1^{\deg_1(v)-\deg_1(u)} \notin \mathcal{M}(u, U)$. Thus, for $i = 1$ $u >_{lex} v$.

Let now $i > 1$ and $0 \leq j < i$ will be the minimal such that $\deg_j(u) < \deg_j(v)$. Then $x_j^{d_j} \in NMP(u, U)$ where $0 < d_j \leq \deg_j(v) - \deg_j(u)$. Since $(v \cdot p)/u$ is multiple of such $x_j^{d_j}$, u cannot $\mathcal{J}-$divide $v \cdot p$. Thereby, both u and v belong to the same monomial group $[d_0, \ldots, d_{i-1}]$. Then one can apply the same reasoning as for $i = 1$ to show that $\deg_i(u) > \deg_i(v)$. □

4 Algorithmic Properties

The following theorem gives an algorithmic characterization of completeness for Janet-like division much like that for Janet and other involutive divisions. Thereby, it establishes a property of Janet-like division which we shall call *continuity* by analogy with that for involutive divisions [1].

Theorem 1. *(Continuity). A monomial set U is $\mathcal{J}-$complete iff*

$$\forall u \in U,\ \forall p \in NMP(u, U),\ \exists v \in U \ : \ v \mid_{\mathcal{J}} u \cdot p. \tag{8}$$

Proof. (5) $\Longrightarrow$ (8) is trivial since the equality (5) is equivalent to

$$\forall u \in U,\ \forall t \in \mathbb{M},\ \exists v \in U \ : \ v \mid_{\mathcal{J}} u \cdot t\,.$$

(8) $\Longrightarrow$ (5) Consider $u \cdot t$ with $u \in U$ and $t \in \mathcal{NM}(u, U)$ as defined in (4). Then $\exists q_1 \in NMP(u, U) \ : \ q_1 \mid t$. From (8) it follows that $\exists u_1 \in U \ : \ u_1 \mid_{\mathcal{J}} q_1$. By Lemma 1, $u_1 \succ_{lex} u$. Thus we have $u \cdot t = u_1 \cdot t_1$. If $t_1 \in \mathcal{M}(u_1, U)$ we are done. Otherwise, $\exists q_2 \in NMP(u, U) \ : \ q_2 \mid t_1$. Again we deduce that $u_1 \cdot t_1 = u_2 \cdot t_2$ where $u_2 \mid_{\mathcal{J}} q_2$ and $u_2 \succ_{lex} u_1$. Repeating this reasoning we obtain the chain of elements in U satisfying

$$u \cdot t = u_1 \cdot t_1 = u_2 \cdot t_2 = \cdots$$

and, by Lemma 1, such that

$$u \prec_{lex} u_1 \prec_{lex} u_2 \prec_{lex} \cdots$$

Since U is finite, the last chain is terminated with a $\mathcal{J}-$divisor of $u \cdot t$. □

Show that Janet-like division satisfies also a property which is the straightforward analogue of property 3 in Definition 4.

Proposition 3. *For every $U \subset \mathbb{M}$ and $u \in U$, the set of $\mathcal{J}-$nonmultipliers $\mathcal{NM}(u, U)$ introduced in Definition 7 satisfies the condition*

$$\forall v \in \mathbb{M} \ : \ \mathcal{NM}(u, U \cup \{v\}) \supseteq \mathcal{NM}(u, U). \tag{9}$$

Proof. From Definition 6 it follows that $NMP(u, U) \neq NMP(u, U \cup \{v\})$ only if there is $1 \leq i \leq n$ such that $u, v \in [d_0, \ldots, d_{i-1}]$ and $\deg_i(v) > \deg_i(u)$. Now, if $\deg_i(u) = \max\{\deg_i(w) \mid w \in [d_0, \ldots, d_{i-1}]\}$ we have

$$NMP(u, U \cup \{v\}) = NMP(u, U) \cup \{x_i^{\deg_i(v)-\deg_i(u)}\}\,,$$

and, hence, $\mathcal{NM}(u, U) \subset \mathcal{NM}(u, U \cup \{v\})$.

Next, if $\deg_i(v) < k_i$ where k_i defined as in Definition 6, then

$$NMP(u, U \cup \{v\}) = NMP(u, U) \cup \{x_i^{\deg_i(v)}\} \setminus \{x_i^{k_i}\}.$$

This implies again $\mathcal{NM}(u, U) \subset \mathcal{NM}(u, U \cup \{v\})$.

At last, if $\deg_i(v) \geq k_i$ the enlargement of U with v does not change the set $NMP(u, U)$. □

Definition 9. (*Completion*). For a given set U, a $\mathcal{J}-$complete set $\bar{U}$ will be called *$\mathcal{J}-$completion* of U if $U \subseteq \bar{U}$ and

$$C_{\mathcal{J}}(\bar{U}) = C(U)\,. \tag{10}$$

Definition 10. (*Prolongation*). The product $u \cdot v$ where $u \in U \subset \mathbb{M}$ and $v \in NMP(u, U)$ will be called a *nonmultiplicative prolongation* of u. Similarly, the product $u \cdot v$ with $v \in \mathcal{M}(u, U)$ will be called a *multiplicative prolongation* of u.

Remark 2. Exactly as in the theory of involutive polynomial bases [1], the above defined notion of prolongation is extended to polynomial sets. Given a monomial order $\succ$ and a polynomial set $F \subset \mathbb{R}$, the product $p \cdot v$ is called nonmultiplicative (multiplicative) prolongation of p if it is such for $\mathrm{lm}(p) \in \mathrm{lm}(F)$.

The following theorem extends the property of constructivity of Janet division [1] to Janet-like division.

Theorem 2. *(Constructivity). A nonmultiplicative prolongation $u \cdot t$, $u \in U, t \in NMP(u, U)$ satisfying $u \cdot t \notin C_{\mathcal{J}}(U)$ and*

$$\forall v \in U,\ \forall s \in NMP(v, U) \text{ such that } v \cdot s \sqsubset u \cdot t \ : \ v \cdot s \in C_{\mathcal{J}}(U) \tag{11}$$

belongs to $\bar{U}$, a $\mathcal{J}-$completion of U.

Proof. Assume for a contradiction that $u\cdot t \notin \bar{U}$. This implies $\exists v \in \bar{U} \ : \ v \mid_{\mathcal{J}} u\cdot t$. From Proposition 3 it follows that $v \notin U$ and $u\cdot t \in \mathcal{M}(v, U\cup\{v\})$. Then $v \sqsubset u\cdot t$ and $\exists u_1 \in U \wedge \exists v_1 \in \mathcal{M}(u_1,U) \ : \ v = u_1v_1$. Since extension of a monomial set U with $\mathcal{J}-$multiplicative prolongation v of u_1 does not affect the group partition (3) of elements in U, it follows that $u\cdot t = u_1\cdot w_1$ where $w_1 \in \mathcal{M}(u_1,U)$ and $w_1 \notin \mathcal{M}(u_1,\bar{U})$. We deduce that $\exists t_1 \in NMP(u_1,\bar{U}) \ : \ t_1 \mid w_1$. Completeness of $\bar{U}$ implies $\exists u_2 \in \bar{U} \ : \ u_2 \mid_{\mathcal{J}} u_1\cdot t_1$. In the obtained equality

$$u\cdot t = u_1\cdot w_1 = u_2\cdot w_2$$

$u_2 \succ_{lex} u_1 \succ_{lex} u$, by Lemma 1. Again, $\exists t_2 \in NMP(u_2,\bar{U}) \ : \ t_2 \mid w_2$, and $\exists u_3 \in \bar{U} \ : \ u_3 \mid_{\mathcal{J}} u_2\cdot t_2$. By repetition of this reasoning, we obtain the infinite chain of elements in $\bar{U}$ satisfying

$$u \prec_{lex} u_1 \prec_{lex} u_2 \prec_{lex} u_3 \prec_{lex} \cdots ,$$

a contradiction. □

The next important property of Janet division - Noetherianity [1,9] is also easily extended to Janet-like division.

Theorem 3. *(Noetherianity). Every set $U \in \mathbb{M}$ admits a $\mathcal{J}-$completion.*

Proof. Follows trivially from the observation that every Janet complete set is also a Janet-like complete, and from Noetherianity of Janet division [1]. The observation is a consequence of Definitions 5, 6 and Janet conditions of completeness. Indeed, for a Janet complete set all the $\mathcal{J}-$nonmultiplicative power products are just $J-$nonmultiplicative indeterminates. An explicit completion of a set U is as follows

$$\bar{U} := \{u\cdot x_1^{i_1}\cdots x_n^{i_n} \mid u\in U,\, 0\le i_j \le h_j(u,U),\, 1\le j\le n\}\,, \qquad (12)$$

where $h_j(u,U)$ are as in Definition 6. Set $\bar{U}$ is both $\mathcal{J}$- and $J-$complete since $\forall v \in \bar{U} \ : \ h_i(v,\bar{U}) \le 1$, and if $h_i(v,\bar{U}) = 1$, then $\exists w \in \bar{U} \ : \ w = v\cdot x_i$. □

Now we are going to show that among different $\mathcal{J}$-complete sets generating the same monomial ideal in $\mathbb{R}$ there is minimal such set.

Proposition 4. *Let $U, V \subset \mathbb{M}$ be $\mathcal{J}$-complete sets such that* $\mathrm{Id}(U) = \mathrm{Id}(V)$. *Then the set $W := U\cap V$ is also $\mathcal{J}$-complete.*

Proof. Since both sets U and V generate the same monomial ideal, W also generates this ideal and U, V are $\mathcal{J}-$completions of W. Assume that W is not $\mathcal{J}$-complete. This implies $\exists u \in W, t \in NMP(u,W) \ : \ u\cdot t \notin C_{\mathcal{J}}(W)$. Choose such a pair of u, t without proper divisors among all other nonmultiplicative prolongations that are not in $C_{\mathcal{J}}(W)$. Then Theorem 2 asserts that $u\cdot t$ must belong to any $\mathcal{J}$-completion of W. Thus, $u\cdot t \in U$ and $u\cdot t \in V$, a contradiction. □

As an immediate consequence of Proposition 4 we have the following result.

Corollary 1. *Given a monomial set U, there exists a minimal $\mathcal{J}$-completion of U, that is, such that it is a subset of any other $\mathcal{J}$-completion of U.*

The next definition is used in [12] for defining Janet-like bases and for proving correctness of the underlying algorithm.

Definition 11. *(Compactness).* A monomial set U will be called $\mathcal{J}-$*compact* if $U \subseteq V$ where V is a minimal $\mathcal{J}-$completion of the reduced Gröbner basis of $\mathrm{Id}(U)$.

5 Conclusion

In this paper we give an explicit receipt of improvement of Janet division. Though the improvement breaks the properties of Janet division as involutive one, the new division called Janet-like inherits all the attractive features of Janet division. At the same time the new division decreases Gröbner redundancy of the output bases.

Recently we analyzed [9] some issues of practical superiority of our Janet division algorithms over the Buchberger algorithm. All those issues are apparently preserved by Janet-like division too. In particular, the new division also admits a tree structure providing a very fast search of Janet-like divisor. Moreover, trees for the new division are more compact than Janet trees.

Our Janet-like division algorithm in its simplest form presented in [12] together with some examples demonstrating its superiority over Janet division algorithms.

We are planning to experiment on Janet-like division to find heuristically best strategies for selection of nonmultiplicative prolongations. For the Buchberger algorithm such a strategy is well-known [13]. For Janet division we already detected some good strategies mentioned in [9]. However, a strategy being good for Janet division may be not always that good for Janet-like division, since the latter provides generally another sequence of intermediate reduction than the former. This important aspect needs further experimental study.

We see also another, pure theoretical, direction of research. Namely, to formulate general properties of a restricted monomial division providing its algorithmic applicability to construction of Gröbner bases. We expect that axioms for involutive division given in Definition 4 or in [2] can be properly modified to characterize such good, though noninvolutive, divisions as Janet-like.

Acknowledgements

The first author thanks A.S.Semenov for useful remarks relating to the proof of Theorem 2. The research presented in this paper was partially supported by the grants 04-01-00784 and 05-02-17645 from the Russian Foundation for Basic Research and the grant 2339.2003.2 from the Russian Ministry of Science and Education.

References

1. Gerdt, V.P. and Blinkov,Yu.A.: Involutive Bases of Polynomial Ideals. *Mathematics and Computers in Simulation* 45 (1998) 519–542, http://arXiv.org/math.AC/9912027; *Minimal Involutive Bases.* Ibid., 543–560, http://arXiv.org/math.AC/9912029.
2. Apel, J.: Theory of Involutive Divisions and an Application to Hilbert Function Computations. *Journal of Symbolic Computation* 25 (1998) 683–704.
3. Janet, M.: *Leçons sur les Systèmes d'Equations aux Dérivées Partielles*, Cahiers Scientifiques, IV, Gauthier-Villars, Paris (1929).
4. Gerdt, V.P., Blinkov,Yu.A. and Yanovich, D.A.: Construction of Janet Bases. I. Monomial Bases. In: *Computer Algebra in Scientific Computing / CASC'01*, V.G.Ganzha, E.W.Mayr and E.V.Vorozhtsov (eds.), Springer, Berlin (2001) 233–247; II. Polynomial bases, ibid., 249–263.
5. Berth, M. and Gerdt, V. Computation of Involutive Bases with Mathematica. In: *Proceedings of the Third International Workshop on Mathematica System in Teaching and Research*, Institute of Mathematics & Physics, University of Podlasie (2001) 29–34.
6. Hausdorf, M. and Seiler, W.M.: Involutive Bases in MuPAD - Part I: Involutive Divisions. *mathPAD* 11 (2002) 51–56.
7. Blinkov, Yu.A., C.F.Cid, C.F., Gerdt, V.P., Plesken, W. and Robertz, D.: The Maple Package "Janet": I.Polynomial Systems. In: *Computer Algebra in Scientific Computing / CASC 2003*, V.G.Ganzha, E.W.Mayr and E.V.Vorozhtsov (eds.). Institute of Informatics, Technical University of Munich, Garching (2003)31–40; II. Linear Partial Differential Equations, ibid., 41–54.
8. Hemmecke, R: *Involutive Bases for Polynomial Ideals.* PhD Thesis, RISC Linz, 2003.
9. Gerdt, V.P.: Involutive Algorithms for Computing Gröbner Bases. IIn "Computational Commutative and Non-Commutative algebraic geometry", S.Cojocaru, G.Pfister and V.Ufnarovski (Eds.), NATO Science Series, IOS Press, 2005, pp.199–225. http://arXiv.org/math.AC/0501111.
10. Buchberger, B.: Gröbner Bases: an Algorithmic Method in Polynomial Ideal Theory, In: *Recent Trends in Multidimensional System Theory*, N.K. Bose (ed.), Reidel, Dordrecht (1985) 184–232.
11. Gerdt, V.P. and Blinkov,Yu.A.: Janet Bases of Toric Ideals. In *Proceedings of the 8th Rhine Workshop on Computer Algebra*, H.Kredel and W.K.Seiler (Eds.), University of Mannheim (2002) 125-135. http://arXiv.org/math.AC/0501180.
12. Gerdt, V.P. and Blinkov,Yu.A.: Janet-like Gröbner Bases. Submitted to CASC'05 (Kalamata, Greece, September 12 - 16, 2005).
13. Giovinni, A., Mora, T., Niesi, G., Robbiano, L. and Traverso, C.: One sugar cube, please, or selection strategies in the Buchberger algorithm. In: *Proceedings of ISSAC'91*, ACM Press, New York (1991) 49–54.

Janet-Like Gröbner Bases

Vladimir P. Gerdt[1] and Yuri A. Blinkov[2]

[1] Laboratory of Information Technologies,
Joint Institute for Nuclear Research, 141980 Dubna, Russia
gerdt@jinr.ru

[2] Department of Mathematics and Mechanics,
Saratov University, 410071 Saratov, Russia
BlinkovUA@info.sgu.ru

Abstract. We define a new type of Gröbner bases called Janet-like, since their properties are similar to those for Janet bases. In particular, Janet-like bases also admit an explicit formula for the Hilbert function of polynomial ideals. Cardinality of a Janet-like basis never exceeds that of a Janet basis, but in many cases it is substantially less. Especially, Janet-like bases are much more compact than their Janet counterparts when reduced Gröbner bases have "sparce" leading monomials sets, e.g., for toric ideals. We present an algorithm for constructing Janet-like bases that is a slight modification of our Janet division algorithm. The former algorithm, by the reason of checking not more but often less number of nonmultiplicative prolongations, is more efficient than the latter one.

1 Introduction

In [1] we introduced the concept of noninvolutive monomial division called Janet-like due to its similarity to Janet division studied in [2]. Having possessed all merits of the latter division, the former division is algorithmically better for constructing Gröbner bases. This is because every Janet divisor is also a Janet-like divisor, and the converse may not hold.

We refer to paper [1] for the basic notations and definitions including those related to Janet-like division and its properties. In the present paper we define Janet-like bases and show that their Gröbner redundancy never exceeds that of Janet bases, but in some cases is considerably less. This effect is illustrated by a number of examples, including toric ideals, which are "unconvenient" for Janet division. We present also the underlying algorithm in its simplest form that is a straightforward modification of our involutive algorithm [3].

2 Janet-Like Bases

In this section we introduce Janet-like bases for polynomial ideals in accordance with general Definition 3 of $r-$bases in paper [1] specified for Janet-like division. However, unlike our more general definition of involutive bases [2], we restrict ourselves to consideration of minimal bases only.

First, we define the corresponding Janet-like reduction and normal form.

Definition 1. ($\mathcal{J}-$*reduction*). Given a monomial order $\succ$, a finite set $F \in \mathbb{R} \setminus \{0\}$ of polynomials and a polynomial $p \in \mathbb{R} \setminus \{0\}$, we shall say that:

(i). $p \in \mathbb{R}$ is $\mathcal{J}-$*reducible modulo* $f \in F$ if p has a term $t = a\,u$ $(a \in \mathbb{K}, u \in \mathbb{M}, a \neq 0)$ whose monomial u is $\mathcal{J}$-multiple[1] of $\mathrm{lm}(f)$. It yields the $\mathcal{J}-$*reduction* $p \rightarrow g := p - (a/lc(f))\, f \cdot v$ where v is $\mathcal{J}-$multiplier for u $(v \in \mathcal{M}(\mathrm{lm}(f), \mathrm{lm}(F)))$.

(ii). p is $\mathcal{J}-$*reducible modulo* F if there is $f \in F$ such that p is $\mathcal{J}-$reducible modulo f.

(iii). p is *in* $\mathcal{J}-$*normal form modulo* F (denotation $p = NF_{\mathcal{J}}(p, F)$) if p is not $\mathcal{J}-$reducible modulo F.

It follows that the normal form $NF_{\mathcal{J}}(g, F)$ $(g \in \mathbb{R})$ can be presented as the finite sum

$$h := NF_{\mathcal{J}}(g, F) = g - \sum_{i=1}^{\mathrm{card}(F)} f_i \sum_j \alpha_{ij} m_{ij}\,, \tag{1}$$

where $\forall i, j \;:\; \alpha_{ij} \in \mathbb{K}$, $m_{ij} \in \mathcal{M}(\mathrm{lm}(f_i), \mathrm{lm}(F))$, $\mathrm{lm}(f_i) m_{ij} \preceq \mathrm{lm}(p)$, $m_{ij} \neq m_{ik}$, $(j \neq k)$, and polynomial h is $\mathcal{J}-$irreducible modulo F.

Definition 2. ($\mathcal{J}-$*autoreduction*). A polynomial set F will be called $\mathcal{J}-$*autoreduced* if

1. The leading monomial set $\mathrm{lm}(F)$ contains distinct elements.
2. Every $f \in F$ has no (tail) terms $t = a\,u$ $(0 \neq a \in \mathbb{K}, u \in \mathbb{M}, u \neq lm(f))$ $\mathcal{J}-$reducible modulo F.

Now we can define Janet-like bases.

Definition 3. (*Janet-like basis*). Let $\mathcal{I} \subset \mathbb{R}$ be a nonzero ideal and $\succ$ be a monomial order. Then a finite $\mathcal{J}$-autoreduced subset $G \subset \mathbb{R}$ such that $\mathcal{I} = Id(G)$ is called *Janet-like basis or* $\mathcal{J}-$*basis of* $\mathcal{I}$ if

$$\forall f \in \mathcal{I},\; \exists g \in G \;:\; \mathrm{lm}(g) \mid_{\mathcal{J}} \mathrm{lm}(f)\,, \tag{2}$$

and set $\mathrm{lm}(G)$ is $\mathcal{J}-$compact in accordance with Definition 11 in [1].

Theorem 1. *(Existence). A Janet-like basis exists for any nonzero ideal $\mathcal{I} \subseteq \mathbb{R}$ and for any admissible monomial order.*

Proof. Let G be a reduced Gröbner basis of $\mathcal{I}$. Let $U := \mathrm{lm}(G)$ be the leading monomial set of G. By Corollary 1 in [1], there exists a minimal $\mathcal{J}-$ completion $\bar{U} \supseteq U$ of U.

If $\bar{U} = U$, then G is also a Janet-like basis. First, it is $\mathcal{J}-$autoreduced, since it is conventionally autoreduced. Second, in accordance to conditions (2) and (10) of paper [1], $\mathrm{lm}(f) \in C_{\mathcal{J}}(U)$ for all $f \in \mathcal{I}$.

[1] As noted in Remark 2 of paper [1], $\mathcal{J}-$division for F is defined in terms of the monomial set $\mathrm{lm}(F)$.

Otherwise, consider $V := \bar{U} \setminus U$. For every $v \in V$ there is $u \in U$ such that $v = u \cdot w$. Note, that w belongs to the monoid ideal $\mathcal{NM}(u, U)$ defined in (4) of paper [1]. For every such v, u, w enlarge G with $g \cdot w$ ($g \in G, \mathrm{lm}(g) = u$). Denote the enlarged set by G_1. Now, if a tail term in G_1 is $\mathcal{J}$-reducible modulo G_1, then perform its $\mathcal{J}$-reduction as described in part (i) of Definition 1. This reduction process obviously terminates in a finite number of step, and we obtain $\mathcal{J}$-reduced set $\bar{G}$ such that $\mathrm{lm}(\bar{G}) = \mathrm{lm}(G_1) = \bar{U}$. Since, by the construction, $\{\mathrm{lm}(f) \mid f \in \mathcal{I}\} = C(U) = C_{\mathcal{J}}(\bar{U})$, the obtained set $\bar{G}$ is a Janet-like basis. □

From the above proof we immediately have the next result just as in theories of Gröbner bases [4] and involutive bases [2,3,5].

Corollary 1. *Given an ideal $\mathcal{I}$ and a monomial order, the following is equivalent:*

(i). G is a Janet-like basis of $\mathcal{I}$.
(ii). G is $\mathcal{J}-$autoreduced, the set $\mathrm{lm}(G)$ is $\mathcal{J}-$compact and

$$\forall f \in \mathcal{I} \,:\, NF_{\mathcal{J}}(f, G) = 0\,. \tag{3}$$

Remark 1. The above proof contains, in fact, one of possible algorithms for constructing Janet-like bases via reduced Gröbner bases. This algorithm, however, needs an algorithm for construction of the reduced Gröbner basis. Below we present another algorithm based on the characterization 3 which computes also reduced Gröbner bases as subsets of Janet-like bases.

As any $r-$basis, a Janet-like basis is a Gröbner basis since $\mathcal{J}$-reducibility implies the conventional reducibility (i.e. reducibility with respect to the conventional division). But the converse is not true in general. By this reason, Janet-like bases, similarly to involutive bases, are generally redundant as the Gröbner one.

The following corollary establishes interrelation between (minimal) Janet, Janet-like and reduced Gröbner bases.

Corollary 2. *Given a minimal Janet basis (abreviation JB), a Janet-like (abreviation JLB) and a reduced Gröbner basis (abreviation GB) of the same ideal, their cardinalities satisfy inequality*

$$\mathrm{card}(GB) \leq \mathrm{card}(JLB) \leq card(JB)\,. \tag{4}$$

Moreover, if all these bases are monic than

$$GB \subseteq JLB \subseteq JB\,. \tag{5}$$

The strict inequalities in (4) and, repectively, strict inclusions in (5) also take place for some ideals and orders.

Proof. If one considers the leading monomial sets of the three bases, then inequality (4) follows from Proposition 1 in [1] and from already shown fact that both Janet and Janet-like bases are Gröbner bases. The proof of inclusion $GB \subseteq JLB$ is contained in the proof of Theorem 1. As to inclusion $JLB \subseteq JB$, it is an easy consequence the same Proposition 1 and of the fact that Janet division satisfies [2] to property 3 in Definition 4 of paper [1]. The last implies that $\mathcal{J}-$autoreduced elements of a Janet-like basis cannot become Janet reducible after extension (completion) of the Janet-like basis to the Janet basis. At last, we illustrate below the strict inequalities and inclusions by some explicit examples. □

Example 1. Consider ideal $\mathrm{Id}(\{x^6y^3-y, x^3y^4-y\}) \in \mathbb{Q}[x,y]$. Its lexicographical $(x \succ y)$ bases are

$$JLB = GB = \{x^3y-y^2, y^5-y\}\,, \quad JB = \{x^3y-y^2, xy^5-xy, x^2y^5-x^2y, y^5-y\}\,.$$

3 Algorithm

In this section we present the simplest version of an algorithm for constructing Janet-like polynomial bases and illustrate its work by Example 1. The algorithm is a straightforward modification of its involutive counterpart [3] and based on the below theorem that gives an algorithmic characterization of Janet-like bases.

To prove the theorem we need the following lemma.

Lemma 1. *For any $\mathcal{J}-$autoreduced polynomial set F, the $\mathcal{J}-$normal form satisfies the linearity condition*

$$\forall p_1, p_2 \in \mathbb{R} \setminus \{0\} \;:\; NF_{\mathcal{J}}(p_1+p_2, F) = NF_{\mathcal{J}}(p_1, F) + NF_{\mathcal{J}}(p_2, F)\,, \qquad (6)$$

Proof. First, we claim that $NF_{\mathcal{J}}(p, F) = 0$ iff p admits representation as a finite sum of the form

$$p = \sum_{i=1}^{\mathrm{card}(F)} f_i \sum_j \beta_{ij} m_{ij} f_j\,, \qquad (7)$$

where $\beta_{ij} \in \mathbb{K}$, $m_{ij} \in \mathcal{M}(\mathrm{lm}(f), \mathrm{lm}(F))$, $m_{ij} \neq m_{ik}$, $(j \neq k)$. If $NF_{\mathcal{J}}(p, F) = 0$, then applying a sequence of elementary $\mathcal{J}-$reduction given in Definition 1, which is obviously terminates by admissibility of order $\succ$, we obtain representation (7) for p. Note, that Proposition 2 in [1] asserts uniqueness of every elementary reduction. Apparently, this implies uniqueness of the representation[2].

Let now $p_3 := p_1 + p_2$ and $h_1 := NF_{\mathcal{J}}(p_1, F)$, $h_2 := NF_{\mathcal{J}}(p_3, F)$, $h_3 := NF_{\mathcal{J}}(p_3, F)$. Then, by Definition 1, $NF_{\mathcal{J}}(h_3 - h_1 - h_2, F) = h_3 - h_1 - h_2$ since h_1, h_2, h_3 have no terms whose monomials belong to $C_{\mathcal{J}}(\mathrm{lm}(F))$. On the other hand, from (1) it follows that $p := h_3 - h_1 - h_2$ admits representation (7). Thus, $NF_{\mathcal{J}}(h_3 - h_1 - h_2, F) = 0$. □

[2] It implies also uniqueness of (1) exactly as in the case of involutive divisions [2].

Algorithm: Janet-like Basis $(F, \prec)$

```
Input: $F \in \mathbb{R} \setminus \{0\}$, a finite set; $\prec$, an order
Output: $G$, a Janet-like basis of $\mathrm{Id}(F)$
 1: choose $f \in F$ with the lowest $\mathrm{lm}(f)$ w.r.t. $\succ$
 2: $G := \{f\}$
 3: $Q := F \setminus G$
 4: do
 5:    $h := 0$
 6:    while $Q \neq \emptyset$ and $h = 0$ do
 7:       choose $p \in Q$ with the lowest $\mathrm{lm}(p)$ w.r.t. $\succ$
 8:       $Q := Q \setminus \{p\}$
 9:       $h :=$ NormalForm$(p, G, \prec)$
10:    od
11:    if $h \neq 0$ then
12:       for all $\{g \in G \mid \mathrm{lm}(h) \sqsubset \mathrm{lm}(g)\}$ do
13:          $Q := Q \cup \{g\};\ G := G \setminus \{g\}$
14:       od
15:       $G := G \cup \{h\}$
16:       $Q := Q \cup \{\, g \cdot t \mid g \in G,\ t \in NMP(\mathrm{lm}(g), \mathrm{lm}(G))\,\}$
17:    fi
18: od while $Q \neq \emptyset$
19: return $G$
```

Theorem 2. *(Algorithmic characterization). An $\mathcal{J}-$autoreduced set $F \in \mathbb{R}$ satisfies (3) for $\mathcal{I} = Id(F)$ iff*

$$\forall f \in F \;\; \forall p \in NMP(\mathrm{lm}(f), \mathrm{lm}(F)) : \; NF_{\mathcal{J}}(f \cdot p, F) = 0\,. \tag{8}$$

Proof. Implication (3) $\Longrightarrow$ (8) is obvious.

(8) $\Longrightarrow$ (3) By Lemma 1, it suffices to show that

$$\forall u \in \mathbb{M}, \forall f \in F \; : \; NF_{\mathcal{J}}(f \cdot u, F) = 0\,. \tag{9}$$

Assume, without the loss of generality, that all polynomials in F are monic. Then conditions (9) together with Theorem 1 in [1] imply $\mathcal{J}-$completeness of $\mathrm{lm}(F)$. Thus, $f \cdot u$ can be rewritten as

$$f \cdot u = g \cdot v + \sum_{i=1}^{\mathrm{card}(F)} f_i \sum_j v_{ij}\,, \tag{10}$$

where $g \in F$ is uniquely defined by f and u, $v \in \mathcal{M}(\mathrm{lm}(g), \mathrm{lm}(F))$, $v_{ij} \in \mathbb{M}$, and $\forall i, j \; : \; lm(f)u = lm(g)v \succ lm(f_i)v_{ij}$. Similarly, we can further rewrite every $f_i \cdot v_{ij}$ in (10) until we obtain for the right-hand side of (10) representation (7). Admissibility of $\succ$ provides termination of this rewriting procedure. □

The above algorithm is an adaptation of our general involutive division algorithm [3] to Janet-like division. Its input consists of a polynomial set F and a monomial order $\succ$. To output a minimal and $\mathcal{J}-$autoreduced set, in accordance to Definition 2, the intermediate polynomial data are separated into subsets G and Q.

Set G contains a part of the intermediate basis. It is initialized at step 2 as a set with the single element $f \in F$ selected at step 1. The rest of the input basis is contained in the set Q initialized at step 3 as $F \setminus \{f\}$.

When the outer **do-while** loop 4-18 is executed, set Q can be enlarged with some elements of G at step 13 and with nonmultiplicative prolongations of polynomials in G. The algorithm terminates when Q becomes empty during execution of the inner **while** loop that signals that all conditions (8) satisfied, and the last $\mathcal{J}-$normal form h computed at step 9, if nonzero, does not have proper divisors of $\mathrm{lm}(h)$ in $\mathrm{lm}(G)$. This condition is verified at step 12.

The choice made at steps 1 and 7 and execution of the **for** loop 12-13 provide correctness of the algorithm. To show this and to show also the algorithm termination, first, consider subalgorithm **Normal Form**. It is invoked in line 9 of the main algorithm and computes $\mathcal{J}-$normal form in the full correspondence with Definition 1 and formula (1). Its termination is an obvious consequence of that for the conventional reductions [4].

Algorithm: Normal Form$(p, G, \prec)$

```
Input: $p \in \mathbb{R} \setminus \{0\}$, a polynomial; $G \subset \mathbb{R} \setminus \{0\}$, a finite set; $\prec$, an order
Output: $h = NF_{\mathcal{J}}(p, G)$, the $\mathcal{J}-$normal form of $p$ modulo $G$
1: $h := p$
2: while $h \neq 0$ and $h$ has a term $t$ $\mathcal{J}-$reducible modulo $G$ do
3:    take $g \in G$ such that $\mathrm{lm}(g) \mid_{\mathcal{J}} t$
4:    $h := h - g \cdot t / \mathrm{lt}(g)$
5: od
6: return $h$
```

Show now *termination* of algorithm **Janet-like Basis**. By the choice done at steps 1 and 7 and by displacement of elements from G to Q at step 13, the elements in $\mathrm{lm}(G)$ occurring right before execution of step 15 have no proper divisors in $\mathrm{lm}(Q)$. Thereby, when the leading monomial $\mathrm{lm}(p)$ of the nonmultiplicative prolongation $g \cdot t \in Q$ with $(g \in G, t \in NMP(\mathrm{lm}(g), \mathrm{lm}(G))$ chosen at step 7 has no $\mathcal{J}-$divisor in $\mathrm{lm}(G)$, the constructivity property (11) in [1] implies that $\mathrm{lm}(p) = \mathrm{lm}(h)$ belongs to any completion of $\mathrm{lm}(G)$. Noetherianity ascertained by Theorem 3 of paper [1] guarantees termination of this completion process.

There are finitely many cases when an element of the input polynomial set F is selected from Q at step 7. Besides, there can only be a finitely many cases when a $\mathcal{J}-$head reducible polynomial p taken from Q has $0 \neq h = NF_{\mathcal{J}}(p, G)$ computed at step 9. This is because in every such case $\mathrm{lm}(h) \notin C(\mathrm{lm}(G))$.

Indeed, assume that there are $g \in G$ and $v \in \mathcal{NM}(\mathrm{lm}(g), \mathrm{lm}(G))$ satisfying $\mathrm{lm}(g) \cdot v = lm(h)$. In that case all nonmultiplicative prolongations of the form $g \cdot t$ with $t \in NMP(\mathrm{lm}(g), \mathrm{lm}(G))$, $t \mid v$, $\mathrm{lm}(g) \cdot t \notin C_{\mathcal{J}}(\mathrm{lm}(G))$ must be added to Q at step 16 of a previous run of the main loop. But then, since $\mathrm{lm}(h) \prec \mathrm{lm}(p)$, all these prolongations must be selected at step 7 and further processed earlier than p. As a result, the leading monomials of these prolongations must belong to $C_{\mathcal{J}}(\mathrm{lm}(G))$ when p is under processing. However, the same arguments as we used in the proof of Theorem 1 in [1], bring us to a contradiction with the assumption made.

To prove *correctness* of algorithm **Janet-like Basis** it suffices to show that the following is a **do-while** loop invariant:

1. $\mathrm{lm}(G)$ is $\mathcal{J}-$compact,
2. The tail monomials in G are not in $C_{\mathcal{J}}(\mathrm{lm}(G))$.

G trivially satisfies both conditions at the initialization step 2. Suppose that this is true after execution of the **while** loop 6-10, and let $G_1 := G \cup \{h\}$ be a set obtained at step 15.

If $\mathrm{lm}(p) = \mathrm{lm}(h)$, as we already seen, $\mathrm{lm}(G_1)$ is compact. Furthermore, by property (9) in [1], the elements in G remain $\mathcal{J}-$reduced after enlargement of G with h. As to the last polynomial, it is in the normal form modulo G, by its construction at step 9.

Consider now the case when $h \neq 0$ and $\mathrm{lm}(p) \succ \mathrm{lm}(h)$. Let G_0 be the value of the intermediate set G right after execution of the **while** loop 6-10, and G_1 be the set obtained at step 15.

Assume that $\mathrm{lm}(G_1)$ is not compact. Then G_1 has a proper subset $G_2 \subset G_1$ with compact $\mathrm{lm}(G_2)$. Then, for any $f \in G_1$ there exists $g \in G_2$ such that $\mathrm{lm}(g) \mid_{\mathcal{J}} \mathrm{lm}(f)$ with respect to the set G_2. At all that $g \neq h$ in accordance to the displacement condition in line 12. Since $\mathrm{Id}(\mathrm{lm}(G_2) = \mathrm{Id}(\mathrm{lm}(G_1))$, polynomial f might only had been added to G as a result of processing a head irreducible nonmultiplicative prolongation of a polynomial $s \in G_0$ which has been displaced at step 13. In this case, however, polynomial f must be also displaced to Q since $\mathrm{lm}(h) \mid \mathrm{lm}(s) \mid \mathrm{lm}(f)$. The obtained contradiction shows compactness of $\mathrm{lm}(G_1)$.

Similarly, if a tail monomial $u \in \mathbb{M}$ of a polynomial $g \in G_1$ became $\mathcal{J}-$reducible modulo $\mathrm{lm}(G_1)$, then it could happen only if u were a nonmultiplicative prolongation $u = \mathrm{lm}(f) \cdot t \prec \mathrm{lm}(g)$ where polynomial f has been moved from G to Q. But in such a case, by the selection strategy of steps 1 and 7, the prolongation $f \cdot t$ must be processed earlier than the nonmultiplicative prolongation of the element in Q whose processing created g. Then, processing of g would lead to $\mathcal{J}-$reduction of u. Thus, $\mathcal{J}-$reducible tail monomials cannot occur in G_1. As to the polynomial h itself, the impossibility of its tail $\mathcal{J}-$reduction after the displacement follows also from the fact that $\mathrm{lm}(h)$ cannot divide its tail monomials.

As an illustration to the internal processing in algorithm **Janet-like Basis**, Table 1 shows intermediate polynomial data for Example 1. The second column of the table contains elements of set G. Their $\mathcal{J}-$nonmultiplicative powers NMP are shown in the third column. The set Q is given in the fourth column. Rows of the table contain these values obtained at the initialization and after every interation of the **do-while** loop. In this case at steps 1 and 7 we selected the lexicographically smallest elements.

Table 1. Intermediate basis elements for Example 1

Steps of algorithm	Sets G and Q		
	elements in G	NMP	Q
initialization	$x^3y^4 - y$	$-$	$\{x^6y^3 - y\}$
iteration	$x^6y^3 - y$ $x^3y^4 - y$	$-$ x^3	$\{x^6y^4 - x^3y\}$
	$x^3y - y^2$	$-$	$\{x^3y^4 - y, x^6y^3 - y\}$
	$x^3y - y^2$ $y^5 - y$	$-$ x^3	$\{x^3y^5 - x^3y, x^6y^3 - y\}$
	$x^3y - y^2$ $y^5 - y$	$-$ x^3	$\{\ \}$

In spite of often redundancy of Janet-like bases as Gröbner ones, as well as in the case of involutive bases [5,6], just this redundancy provides more accessibility to information on polynomial ideals and modules. In particular, Janet-like bases also give explicit formulae for the Hilbert function (cf. [5]) and Hilbert polynomial (cf. [3]) of a polynomial ideal $\mathcal{I}$ in terms of binomial coefficients. If G is a $\mathcal{J}-$ basis of $\mathcal{I}$, then the (affine) Hilbert function $HF_{\mathcal{I}}(s)$ and the Hilbert polynomial $HP_{\mathcal{I}}(s)$ are

$$HF_{\mathcal{I}}(s) = \binom{n+s}{s} - \sum_{i=0}^{s} \sum_{u\in \mathrm{lm}(G)} \sum_{i_1=0}^{d_1-1} \cdots \sum_{i_k=0}^{d_k-1} \binom{i - \sum_j i_j - deg(u) + \mu(u) - 1}{\mu(u) - 1},$$

$$HP_{\mathcal{I}}(s) = \binom{n+s}{s} - \sum_{u\in \mathrm{lm}(G)} \sum_{i_1=0}^{d_1-1} \cdots \sum_{i_k=0}^{d_k-1} \binom{s - \sum_j i_j - deg(u) + \mu(u)}{\mu(u)}.$$

Here, if $NMP(u, \mathrm{lm}(G)) \neq \emptyset$, then $NMP(u, \mathrm{lm}(G)) := \{x_1^{d_1}, \ldots, x_k^{d_k}\}$ with $d_j \neq 0$ $(1 \leq j \leq k)$ and $\mu(u) := n - k$. Otherwise, $k := 1, d_1 := 0, \mu(u) := n$. The first term in the right hand sides of these formulae is the total number of monomials in $\mathbb{M}$ of degree $\leq s$. The sum in the expression for $HF_{\mathcal{I}}(s)$ counts the number of monomials of degree $\leq s$ in the set $C_{\mathcal{J}}(\mathrm{lm}(G))$ defined in (6) of paper [1]. In accordance to the completeness condition (5) in [1], this number coincides with the number of such monomials in the monoid ideal $C(\mathrm{lm}(G))$ as defined in (7) of paper [1].

4 Illustrative Examples

In this section we consider four more nontrivial examples than small Example 1. Our goal is to compare their Janet-like bases with minimal involutive Janet bases and reduced Gröbner bases. We present these examples in the increasing order with respect to the cardinalities of Janet bases. Two of examples generating toric ideals we took from [7] and [8] and already used in [9] to show limitations in applicability of Janet bases. In the last paper we also shortly noted the approach described in the present paper. One more toric ideal was taken from [10] where it was presented already in the Gröbner basis form. One of the examples [11] is not toric ideal, but also demonstrates deficiency in application of Janet bases to certain problems.

The below examples have compact input and comparatively compact (reduced) Gröbner bases whereas their Janet bases are much larger. For all the examples we used the degree-reverse-lexicographical monomial order induced by the explicitly indicated order on the variables.

Computations were performed with our C++ code implementing Janet division algorithm [3]. We extended the package with our first implementation of Janet-like division. The actual algorithm that has been implemented is an improved version of the above algorithm **Janet-like Basis**. The improvement is similar to that described in [3] for involutive division.

The reduced Gröbner bases given explicitly whereas Janet and Janet-like bases given only for the first rather small example. In addition, for the listed examples we computed their Hilbert polynomials via Janet-like bases (see Sect.5).

Example 2. (*Toric ideal I*) [7] $\{\, x^7-y^2z, x^4w-y^3, x^3y-zw \,\}$ $(x \succ y \succ z \succ w)$.

Gröbner basis: $\{\, x^7-y^2z, x^4w-y^3, x^3y-zw, y^4-xzw^2 \,\}$.

Janet-like basis: $\{\, x^7-y^2z, x^4y-xzw, x^4w-y^3, x^3y-zw, y^4-xzw^2 \,\}$.

Janet basis: $\{\, x^7-y^2z, x^6y-x^3zw, x^6w-x^2y^3, x^5y-x^2zw, x^2y^4-x^3zw^2,$
$x^5w-xy^3, x^4y-xzw, x^2zw^2-xy^4, x^4w-y^3, x^3y-zw, y^4-xzw^2 \,\}$.

Hilbert Polynomial : $\dfrac{39}{6}s^2-\dfrac{21}{2}s+5$.

Example 3. (*Polynomial ideal*) [11] $(w \succ x \succ y \succ z)$

$$\{\, z^{20}+z^{10}-x^2, z^{30}+z^{10}-x\,y^3, w^{40}x^4-y^6 \,\}.$$

Gröbner basis:

$\{\, 16w^{40}z^{10}-16w^{40}x^2+y^{18}-x^{12}+9x^9y^3-24y^{12}-33x^{10}+150x^7y^3+8z^{10}$
$-219x^8+627x^5y^3-470*x^6+690x^3y^3+16y^6-502x^4+188xy^3-196x^2,$
$16w^{40}y^9-16w^{40}x^3-y^{27}+32y^{21}+x^16y^3-12x^17+98x^{14}y^3-374x^{15}+$
$1875x^{12}y^3-160y^{15}-3778x^{13}+13743x^{10}y^3-17179x^{11}+45923x^8y^3$
$-41148x^9+74362x^6y^3+120y^9-57702x^7+60452x^4y^3-1760xy^6-45324x^5$
$+18416x^2y^3-16728x^3$, $w^{40}x^4-y^6$, $8w^{40}xy^3-8w^{40}x^2+y^{18}-x^{12}+9x^9y^3$
$-22y^{12}-33x^{10}+150x^7y^3+8xy^9-221x^8+631x^5y^3-472x^6+688x^3y^3+$
$8y^6-506x^4+192xy^3-192x^2, z^{20}+z^{10}-x^2$, $xy^{12}-x^9+6x^6y^3+2y^9$
$-13x^7+41x^4y^3-8xy^6-24x^5+28x^2y^3-22x^3$, $2y^3z^{10}+2xz^{10}-xy^6+x^5$
$-3x^2y^3, x^2z^{10}+2z^{10}-xy^3-x^2$, $x^2y^6-x^6+3x^3y^3-2x^4+2xy^3-2x^2 \,\}$.

Hilbert Polynomial: $2980\,s-76460$.

Example 4. (*Toric ideal II*) [8,9] $(x_0 \succ x_1 \succ x_2 \succ x_3 \succ x_4)$

$$\{ \; x_0x_1x_2x_3x_4 - 1, x_2^{29}x_3^5 - x_1^{14}x_4^{20}, x_1^{39} - x_2^{25}x_3^{14} \; \} \, .$$

Gröbner basis:

$$\begin{aligned}
\{ \; & x_0{x_1}^2x_3{x_4}^{281} - {x_2}^{280}, {x_2}^{281} - x_1{x_4}^{280}, x_0{x_3}^2{x_4}^{221} - x_1{x_2}^{218}, \\
& {x_1}^2{x_2}^{219} - x_3{x_4}^{220}, x_0{x_3}^3{x_4}^{161} - {x_1}^4{x_2}^{156}, {x_1}^5{x_2}^{157} - {x_3}^2{x_4}^{160}, \\
& x_0{x_3}^4{x_4}^{101} - {x_1}^7{x_2}^{94}, {x_1}^8{x_2}^{95} - {x_3}^3{x_4}^{100}, x_0{x_1}^4{x_4}^{61} - {x_2}^{61}, \\
& {x_2}^{62}x_3 - {x_1}^3{x_4}^{60}, x_0{x_3}^5{x_4}^{41} - {x_1}^{10}{x_2}^{32}, {x_1}^{11}{x_2}^{33} - {x_3}^4{x_4}^{40}, \\
& x_0{x_2}^{26}{x_3}^{15}x_4 - {x_1}^{38}, {x_1}^{39} - {x_2}^{25}{x_3}^{14}, x_0{x_1}^{15}{x_4}^{21} - {x_2}^{28}{x_3}^4, \\
& {x_2}^{29}{x_3}^5 - {x_1}^{14}{x_4}^{20}, x_0{x_3}^{10}{x_4}^{21} - {x_1}^{24}{x_2}^3, {x_1}^{25}{x_2}^4 - {x_3}^9{x_4}^{20}, \\
& x_0x_1x_2x_3x_4 - 1 \; \} \, .
\end{aligned}$$

$$\text{Hilbert Polynomial}: \quad \frac{3905}{2}\,s^2 - \frac{177005}{2}\,s + 178805 \, .$$

Example 5. (*Toric ideal III*) [10] $(x \succ y \succ z \succ w)$

Gröbner basis:

$$\begin{aligned}
\{ \; & y^{250} - x^{239}z^{11}, \; x^{150}z^{12} - y^{161}w, \; y^{89}z - x^{89}w \; x^{61}z^{13} - y^{72}w^2, \\
& x^{33}z^{27} - y^{55}w^5, \; z^{55} - x^{23}y^{21}w^{11}, \; x^5z^{41} - y^{38}w^8, \; y^{17}z^{14} - x^{28}w^3 \; \} \, .
\end{aligned}$$

$$\text{Hilbert Polynomial}: \quad \frac{1229}{2}\,s^2 - \frac{73855}{2}\,s + 546272 \, .$$

Table 2. Cardinalities of bases in Examples 2-5

Example	Cardinality		
	Gröbner basis	Janet-like basis	Janet basis
2	4	5	11
3	9	14	983
4	19	190	7769
5	8	18	37901

In Table 2 we show cardinalities of Gröbner, Janet-like and Janet bases for the above examples. As one sees, Janet-like bases are much more compact than Janet bases. In other words, they have much less Gröbner redundancy. This higher redundancy of Janet bases has an effect on the running times.

The timings for construction of Janet bases for Examples 3, 4 and 5 [3], as measured on a AthlonXP 1600 computer with 256 Mb RAM running under Gentoo Linux 2004.3, are 0.04, 3.73 and 427.52 seconds, respectively. As to the

[3] Example 2 is too small for our code.

timings for Janet-like bases, because of their very small value (certainly less that 0.01 second) we were not able to measure them.

It is clear that for the case of Janet bases those computing times are wasted for constructing, analyzing and adding to the output basis a large number of $J-$head irreducible nonmultiplicative prolongations. Janet-like division is much more optimized in this respect.

5 Conclusion

It should be noted that, given a Janet-like basis G of a polynomial ideal $\mathcal{I} := \mathrm{Id}(G)$, the set

$$\{\ tg \mid g \in G, t \in \mathcal{M}(\mathrm{lm}(g), \mathrm{lm}(G))\ \} \tag{11}$$

can be considered as a staggered linear basis of $\mathcal{I}$ as $\mathbb{K}-$vector space. Similarly, any involutive basis G generates a staggered linear basis if one replaces $\mathcal{M}(\mathrm{lm}(g), \mathrm{lm}(G))$ in (11) by $\mathcal{L}(\mathrm{lm}(g), \mathrm{lm}(G))$ in accordance with Definition 4 in [1].

The notion of staggered linear basis was introduced in [13] (see also [14]) together with the appropriate modification of the Buchberger algorithm for computing Gröbner bases. Based upon relation (11), one can consider algorithm **Janet-like Basis**, as well as involutive algorithms [2,5,3], as improvements of the Gebauer-Möller staggered linear basis algorithm [13]. Another and very efficient imrovement of the last algorithm is the Faugère algorithm F_5 described in [15] for the case of homogeneous input polynomials. The most impressive feature of F_5 is detecting practically all useless, i.e., zero-redundant critical pairs.

The radical distinction of algorithm **Janet-like Basis** and involutive algorithms from the Gebauer-Möller staggered linear basis algorithm and the Faugère algorithm F_5 is partition of monomials for every of intermediate polynomials into two disjoint sets: multiplicative and nonmultiplicative. These two sets play fundamentally different algorithmic role. Whereas nonmultiplicative monomials are used for construction of prolongations including critical pairs, the multiplicative ones are used for reduction only. As a result, both intermediate and output bases generally have some extra Gröbner redundant elements that are nonmultiplicative prolongations of other elements in the basis. In doing so, the reduced Gröbner basis is the internally fixed subset of the output basis, and can be output without any extra computational costs.

On the other hand, experimental study of Janet division presented in [3] shows that the presence of extra polynomials provided by the partition of monomials smoothes growth of intermediate coefficients, and thereby increases practical efficiency of computation. Other efficiency aspects of the partition observed in [3] are: weakened role of the Buchberger criteria, fast search of a reductor, natural and effective parallelism. By its similarity to Janet division, Janet-like division preserves all these efficiency issues. The experimental evidence of this fact will be described elsewhere.

There are grounds to believe that the new criterion of paper [15] can be adopted to our algorithms too.

Acknowledgements

The research presented in this paper was supported in part by the grants 04-01-00784 and 05-02-17645 from the Russian Foundation for Basic Research and the grant 2339.2003.2 from the Russian Ministry of Science and Education.

References

1. Gerdt, V.P. and Blinkov,Yu.A.: Janet-like Monomial Division. Submitted to CASC'05 (Kalamata, Greece, September 12 - 16, 2005).
2. Gerdt, V.P. and Blinkov,Yu.A.: Involutive Bases of Polynomial Ideals. *Mathematics and Computers in Simulation* 45 (1998) 519–542, http://arXiv.org/math.AC/9912027; *Minimal Involutive Bases.* Ibid., 543–560, http://arXiv.org/math.AC/9912029.
3. Gerdt, V.P.: Involutive Algorithms for Computing Gröbner Bases. IIn "Computational Commutative and Non-Commutative algebraic geometry", S.Cojocaru, G.Pfister and V.Ufnarovski (Eds.), NATO Science Series, IOS Press, 2005, pp.199–225. http://arXiv.org/math.AC/0501111.
4. Buchberger, B.: Gröbner Bases: an Algorithmic Method in Polynomial Ideal Theory, In: *Recent Trends in Multidimensional System Theory*, N.K. Bose (ed.), Reidel, Dordrecht (1985) 184–232.
5. Apel, J.: Theory of Involutive Divisions and an Application to Hilbert Function Computations. *Journal of Symbolic Computation* 25 (1998) 683–704.
6. Seiler, W.M. *Involution - The formal theory of differential equations and its applications in computer algebra and numerical analysis*, Habilitation thesis, Dept. of Mathematics, University of Mannheim (2002).
7. Bigatti, A.M., La Scala, R. and Robbiano, L.: Computing Toric Ideals. *Journal of Symbolic Computation* 27 (1999) 351-365.
8. Pottier, L.: *Computation of toric Gröbner bases, Gröbner bases of lattices and integer point sof polytopes.* http://www-sop.inria.fr/safir/SAM/Bastat/doc/doc.html
9. Gerdt, V.P. and Blinkov,Yu.A.: Janet Bases of Toric Ideals. In *Proceedings of the 8th Rhine Workshop on Computer Algebra*, H.Kredel and W.K.Seiler (Eds.), University of Mannheim (2002) 125-135. http://arXiv.org/math.AC/0501180.
10. Morales, M.: Equations des Variétés Monomiales en codimension deaux. *Journal of Algebra* 175 (1995) 1082-1095.
11. Hemmecke, R.: Private communication.
12. Giovinni, A., Mora, T., Niesi, G., Robbiano, L. and Traverso, C.: One sugar cube, please, or selection strategies in the Buchberger algorithm. In: *Proceedings of ISSAC'91*, ACM Press, New York (1991) 49–54.
13. Gebauer, R. and Möller, H.M.: Buchberger's Algorithm and Staggered Linear Bases. In: *Proceedings of SYMSAC '86*, ACM Press, New York (1986) 218–221.
14. Möller, H.M., Mora, T. and Traverso, C.: Gröbner Bases Computation Using Syzygies. In: *Proceedings of ISSAC'92*, ACM Press, New York (1992) 320–328.
15. Faugère, J.C. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In: *Proceedings of Issac 2002*, ACM Press, New York (2002) 75–83.

Circulant Digraphs and Monomial Ideals

Domingo Gómez, Jaime Gutierrez, and Álvar Ibeas

Faculty of Sciences, University of Cantabria,
Santander E–39071, Spain
jaime.gutierrez@unican.es

Abstract. It is known that there exists a Minimum Distance Diagram (MDD) for circulant digraphs of degree two (or double-loop computer networks) which is an L-shape. Its description provides the graph's diameter and average distance on constant time. In this paper we clarify, justify and extend these diagrams to circulant digraphs of arbitrary degree by presenting monomial ideals as a natural tool. We obtain some properties of the ideals we are concerned. In particular, we prove that the corresponding MDD is also an L-shape in the affine r-dimensional space. We implement in PostScript language a graphic representation of MDDs for circulant digrahs with two or three jumps. Given the irredundant irreducible decomposition of the associated monomial ideal, we provide formulae to compute the diameter and the average distance. Finally, we present a new and attractive family (parametrized with the diameter $d > 2$) of circulant digraphs of degree three associated to an irreducible monomial ideal.

1 Introduction

Let Γ be a finite group and $S = \{j_1, \ldots, j_r\}$ a subset of Γ. The *Cayley digraph* of Γ with respect to S is a digraph whose vertex set is Γ and such that (g, h) is a directed edge if and only if $g^{-1}h \in S$. Let N be a positive natural number and $\mathbb{Z}_N$ the integers modulo N; we denote by $C_N(S) = C_N(j_1, \ldots, j_r)$ the Cayley digraph of the cyclic group $\mathbb{Z}_N$. $C_N(j_1, \ldots, j_r)$ is called the circulant digraph of jumps $j_1, \ldots, j_r$. If S is a subset of $\mathbb{Z}_N$ verifying: for every $j \in S$, also $-j \in S$, then $C_N(S)$ is called (undirected) circulant graph. $C_N(j_1, \ldots, j_r)$ is a vertex-symmetric graph of degree r. And it is connected if and only if $\gcd(j_1, \ldots, j_r, N) = 1$.

Circulant digraphs or multi-loop computer networks have a vast number of applications in telecommunication networking, VLSI design and distributed computation. Their properties, such as diameters and reliabilities, have been the focus of many research in computer network design, see for instance [3,5,6,10,11].

The particular case $r = 2$, this is, circulant digraphs of degree two or double-loop networks, has been extensively studied, see the surveys [2,4]. When $C_N(j_1, j_2)$ is connected, we can define a Minimum Distance Diagram (MDD) as an array with vertex 0 in cell $(0, 0)$ and vertex c in cell (x, y) (x is the column and y the row index), for an instance making $j_1x + j_2y \equiv c \bmod N$ and $x + y$ minimum.

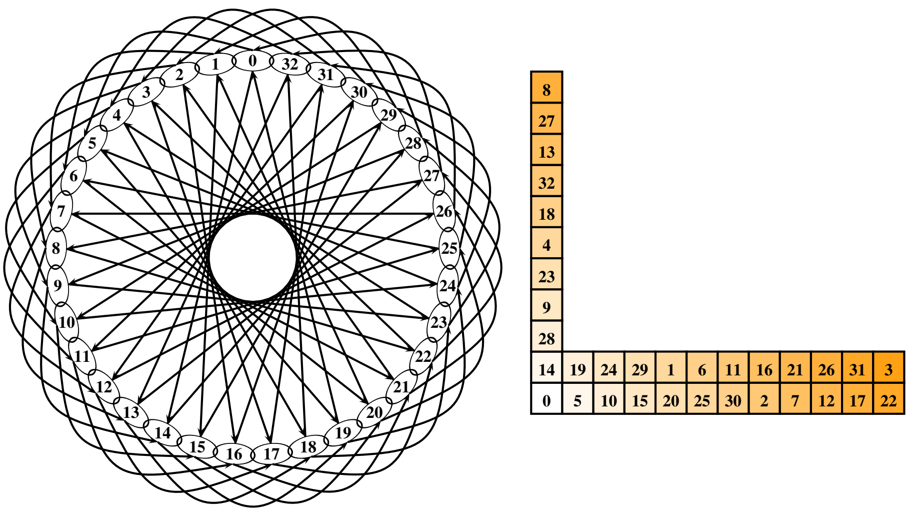


Fig. 1. $C_{33}(5, 14)$

The classical work by Wong and Coppersmith (1974) [10] presents an algorithm to construct an MDD of $C_N(j_1, j_2)$ on $O(N^2)$ steps and showing it is an L-shape. They also gave a general idea to construct an MDD of $C_N(1, s_1, \ldots, s_{r-1})$.

Two notable parameters in a graph are the diameter $d(C_N)$ and the average distance $\bar{d}(C_N)$. The diameter represents the worst delay in the communication between two nodes and the average distance the average delay. Given an L-shape it is easy to recover both quantities.

On the other hand, if we denote by $d_r(N) = \min\{d(C_N(j_1, \ldots, j_r) : j_1, \ldots, j_r \in \mathbb{Z}_N\}$, an important problem in circulant digraph theory is determining $d_r(N)$ and finding $C_N(j_1, \ldots, j_r)$ which attains this minimum diameter. The network $C_N(j_1, \ldots, j_r)$ is said to be *optimal* if $d(C_N) = d_r(N)$. In some cases, it is difficult to obtain such networks; however, one can find general simple functions $lb_r(N)$ and $upb_r(N)$ which are for every N a lower and an upper bound for $d_r(N)$, see [2]. The paper [10] showed $lb_2(N) = \sqrt{3N} - 2$ and presented a family of circulant digraphs having diameter $2\sqrt{N} - 2$.

In this article we present the monomial ideals as a natural tool to study MDDs for general circulant digraphs of arbitrary degree. Given a graded monomial ordering and a circulant digraph $C_N(j_1, \ldots, j_r)$ we build a monomial ideal in the polynomial ring $\mathbb{K}[X_1, \ldots, X_r]$, where $\mathbb{K}$ is an arbitrary field. We obtain some properties of this ideal, in particular, we prove that the corresponding MDD is also an L-shape in the affine r-dimensional space. We implement in PostScript language a graphic representation of a MDD for circulant digraphs of degree two or three. Given a minimal system of generators of the monomial ideal we provide formulae to compute $d(C_N)$ and $\bar{d}(C_N)$ for the corresponding circulant digraph $C_N(j_1, \ldots, j_r)$. We also show a family of circulant digraphs of degree two, which basically coincides with the family obtained in paper [10]. Finally, given a natural number $d > 2$ we build a circulant digraph of degree three with diameter d, average distance $d/2$ and having an irreducible monomial ideal for any graded monomial ordering.

The paper is divided into six sections. In Section 2 we collect several known facts about monomial ideals, presenting examples and fixing notation for later use. Section 3 regards the key idea of associating circulant digraphs to monomial ideals in order to obtain a MDD. Section 4 is dedicated to provide formulae to find the diameter and the average distance. Then (Section 5) we show families of circulant graphs of degree two and degree three. We conclude with a discussion of open questions.

This is an extended abstract. A detailed version may be consulted at `http://personales.unican.es/ibeasaj/circula`.

2 Monomial Ideals

Monomials ideals form an important link between commutative algebra and combinatorics. Here we review several basic related results and definitions concerning monomial ideals, see for instance [1,9]. We will apply these results several times along the paper.

Let $\mathbb{K}$ be an arbitrary field and $\mathbb{K}[X_1, \ldots, X_r]$ the polynomial ring in the variables $X_1, \ldots, X_r$. Throughout the paper, we very often identify monomials of $\mathbb{K}[X_1, \ldots, X_r]$ with vectors of $\mathbb{N}^r$ and we use the following notation:

$$\mathbf{x}^{\mathbf{a}} = X_1^{a_1} \cdots X_r^{a_r} \longleftrightarrow \mathbf{a} = (a_1, \ldots, a_r).$$

$$\mathbf{x}^{\mathbf{a}} | \mathbf{x}^{\mathbf{b}} \iff \mathbf{a} = (a_1, \ldots, a_r) \leq \mathbf{b} = (b_1, \ldots, b_r) \iff \forall i = 1, \ldots, r,\ a_i \leq b_i.$$

$$\mathbf{e}_i = (0, \ldots, \overset{\overset{i}{\smile}}{1}, \ldots, 0).$$

$$\mathfrak{m}^{\mathbf{a}} = (X_i^{a_i} \ / \ a_i > 0)$$

A *monomial ideal* is an ideal generated by monomials, i.e., $I \subset \mathbb{K}[X_1, \ldots, X_r]$ is a monomial ideal if there is a subset $A \subset \mathbb{N}^r$ such that:

$$I = (\mathbf{x}^{\mathbf{a}} \ / \ \mathbf{a} \in A) =: (A).$$

We deal with two ways of describing an arbitrary monomial ideal:

– via the minimal system of generators, $I = (\mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s})$, we have:

$$\mathbf{x}^{\mathbf{u}} \in I \iff \exists\ i \in \{1, \ldots, s\} \ / \ \mathbf{a}_i \leq \mathbf{u},$$

– via the (unique) irredundant irreducible decomposition, $I = \mathfrak{m}^{\mathbf{b}_1} \cap \cdots \cap \mathfrak{m}^{\mathbf{b}_n}$, we have:

$$\mathbf{x}^{\mathbf{u}} \notin I \iff \exists\ i \in \{1, \ldots, n\} \ / \ \mathbf{u} < \widehat{\mathbf{b}}_i, \text{ where } \widehat{b_{i,j}} := \begin{cases} b_{i,j}, & \text{if } b_{i,j} > 0 \\ +\infty, & \text{if } b_{i,j} = 0 \end{cases}.$$

An algorithm for finding a primary decomposition of a monomial ideal is based on the Alexander duality (see [7]). An irreducible component $\mathfrak{m}^{\mathbf{a}}$ can be associated with $\operatorname{lcm}(X_1^{a_1}, \ldots, X_r^{a_r}) = \mathbf{x}^{\mathbf{a}}$. On the other hand, if $\mathbb{K}[X_1, \ldots, X_r]/I$ is an artinian ring then the monomial $\mathbf{x}^{\mathbf{a}}$ associated with the irreducible component $\mathfrak{m}^{\mathbf{a}}$ must coincide with the least common multiple of a subset of the minimal generators of I. The following is a straight-forward result:

Theorem 1. *Let I be a nontrivial monomial ideal given by the minimal system of generators $I = (\mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s})$ and by the irredundant irreducible decomposition $I = \mathfrak{m}^{\mathbf{b}_1} \cap \cdots \cap \mathfrak{m}^{\mathbf{b}_n}$. The following are equivalent:*

1. *$\mathbb{K}[X_1, \ldots, X_r]/I$ is an artinian ring.*
2. *$\forall i = 1, \ldots, r,\ \exists j \in \{1, \ldots, s\}, \exists e \in \mathbb{N} \;/\; \mathbf{a}_j = (0, \ldots, \overset{\overset{i}{\smile}}{e}, \ldots, 0)$.*
3. *$\forall i = 1, \ldots, n,\ \forall j \in \{1, \ldots, r\} \;/\; b_{i,j} > 0$.*

We conclude this section by illustrating these facts by an example:

Example 1. Let I_2 be the following monomial ideal (see Figure 2):

$$I_2 = (x^8, x^4y^2, y^5, y^3z, z^5, x^3z^4, x^7z, x^3y^2z^2) =$$

$$= (x^8, y^2, z) \cap (x^7, y^2, z^4) \cap (x^4, y^3, z^2) \cap (x^4, y^5, z) \cap (x^3, y^3, z^5).$$

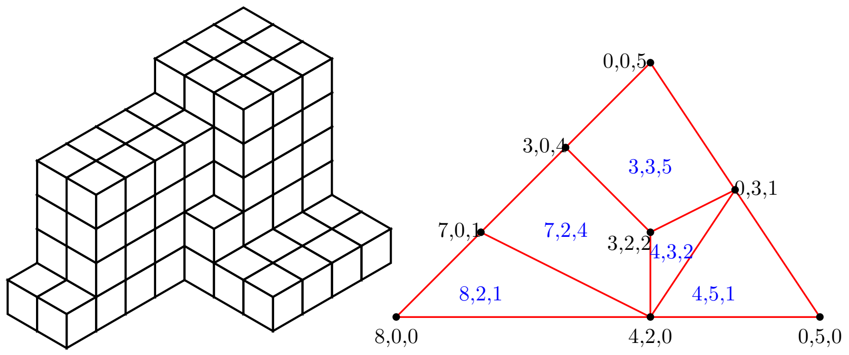


Fig. 2.

In [8], a planar graph is associated to every monomial ideal in three dimensions. Whenever the $\mathbb{K}$-dimension of the ideal is finite, the associated monomial $\mathbf{x}^{\mathbf{b}}$ to an irreducible component $\mathfrak{m}^{\mathbf{b}}$ is identified with a bounded connected component is the graph's complement and can be obtained as the least common multiple of generators in its boundary.

The description of those relations will permit simplifying several computations on the monomial ideals associated with circulant digraphs.

3 Minimum Distance Diagram (MDD)

Monomial ideals also arise in graph theory. Given a graph G with vertices $X_1, \ldots, X_r$; it is associated with the monomial ideal generated by the quadratic monomials X_iX_j such that X_i is adjacent to X_j (see for instance [9]). In this section we propose a different approach to study circulant digraphs: we also associate a circulant digraph with a monomial ideal.

The set of paths can be identified with $\mathbb{N}^r$, being $\mathbf{a} = (a_1, \ldots, a_r)$ the path formed by joining a_i jumps of type j_i, $(1 \leq i \leq r)$. So, each path becomes a monomial and conversely.

Given a connected circulant digraph $C_N(j_1, \ldots, j_r)$ we are looking for a shortest path from node 0 to node c for all $c \in \mathbb{Z}_N$: a minimum distance diagram (MDD). We can construct the *routing function* R:

$$\begin{array}{rcc} R : \mathbb{N}^r & \longrightarrow & \mathbb{Z}_N \\ \mathbf{a} & \rightsquigarrow & a_1j_1 + \cdots + a_rj_r . \end{array}$$

Thus, we need to find a right inverse map of R:

$$D : \mathbb{Z}_N \longrightarrow \mathbb{N}^r ,$$

such that

$$R(D(c)) = c, \ \forall c \in \mathbb{Z}_N \quad \wedge \quad \|D(c)\|_1 = \min\{\|\mathbf{x}\|_1 \ / \ \mathbf{x} \in R^{-1}(c)\}.$$

In general the map D is not unique. This happens when the set $R^{-1}(c)$ contains two or more elements with minimum ℓ_1 norm, for some $c \in \mathbb{Z}_N$.

In circulant digraphs of degreee two we can characterize this situation in terms of lattices. Let $\bar{R}$ be the extended map of R from $\mathbb{N}^r$ to $\mathbb{Z}^r$ and, $\mathcal{L}$ the kernel of the map $\bar{R}$.

Proposition 1. *With the above notation, and given an MDD D, we have that $C_N(j_1, j_2)$ has more than one MDD if and only if there exists a vector $(T, -T) \in \mathcal{L}$ with $T > 0$ and $T \leq \max\{a_1, a_2\}$, for some $\mathbf{a} = (a_1, a_2) \in D(\mathbb{Z}_N)$.*

In the introductory example $C_{33}(5, 14)$:

$$(T, -T) = \alpha(-16, 1) + \beta(-1, -2) \in \mathcal{L} \iff \alpha = \frac{-A}{11}, \ \beta = \frac{5T}{11} \in \mathbb{Z} \iff T \in (11).$$

In consequence, this graph admits exactly four MDDs, the L-shape given in the Introduction section, and the three ones shown in Figure 3.

However, only two of them are L-shapes and they correspond with the only two graded monomial orderings in $\mathbb{K}[X, Y]$. So, according with the previous discussion in order to determine a unique MDD we need a well-order relation in $\mathbb{N}^r$, for instance a monomial ordering. Since the norm ℓ_1 equals the monomial degree, the monomial ordering should be graded. Fixing a graded monomial ordering $\preceq$, then the associated MDD is:

$$\begin{array}{rcc} D : \mathbb{Z}_N & \longrightarrow & \mathbb{N}^r \\ c & \rightsquigarrow & \min(R^{-1}(c)). \end{array}$$

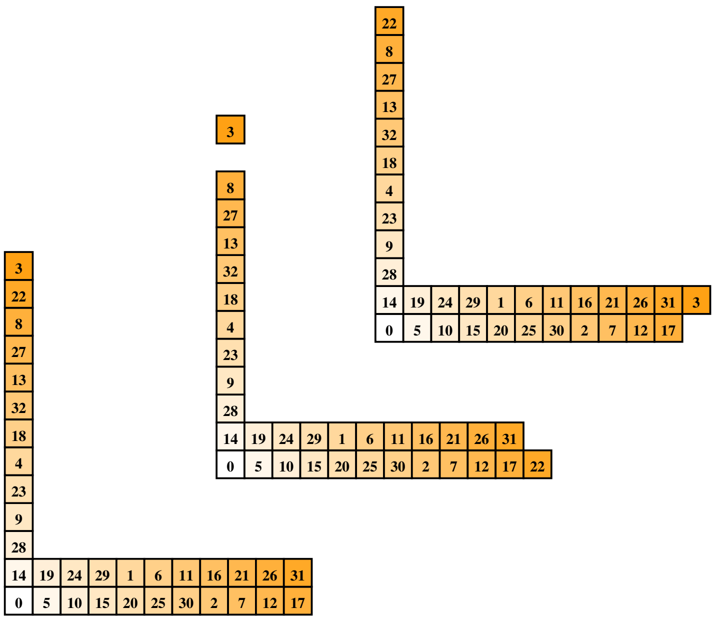


Fig. 3. Different MDD's for the same graph

For each monomial ordering we can associate the bijective map $p : \mathbb{N} \longrightarrow \mathbb{N}^r$, such that $n < m \Rightarrow p(n) \prec p(m)$. That is, verifying

$$p(i) = \min \left(\mathbb{N}^r \backslash \{ p(j) \ / \ j < i \} \right) .$$

This map provides a method for constructing the MDD with respect to the fixed monomial ordering. The method consists on visiting (through p) the points (paths)of $\mathbb{N}^r$ until cover all nodes ($\mathbb{Z}_N$).

We have implemented in PostScript language a graphic representation of the MDD associated to a circulant digraph of degree two or three and a graded lexicographic (reverse or not) ordering. The implementation uses the function s:

$$\begin{array}{rcc} s : \mathbb{N}^r & \longrightarrow & \mathbb{N}^r \\ \mathbf{a} & \rightsquigarrow & p(p^{-1}(\mathbf{a}) + 1) \end{array} .$$

Algorithm 2 (MDD construction). *We have as input the circulant digraph and the map s corresponding with a graded lexicographic monomial ordering $\preceq$:*

$$\mathbf{a} \prec s(\mathbf{a}), \ (\mathbf{a} \prec \mathbf{b} \Rightarrow s(\mathbf{a}) \preceq \mathbf{b})$$

INPUT $j_1, \ldots, j_r, N \in \mathbb{N}, \ gcd(j_1, \ldots, j_r, N) = 1; \ s$
OUTPUT $D(c), \ c = 0, \ldots, N-1$

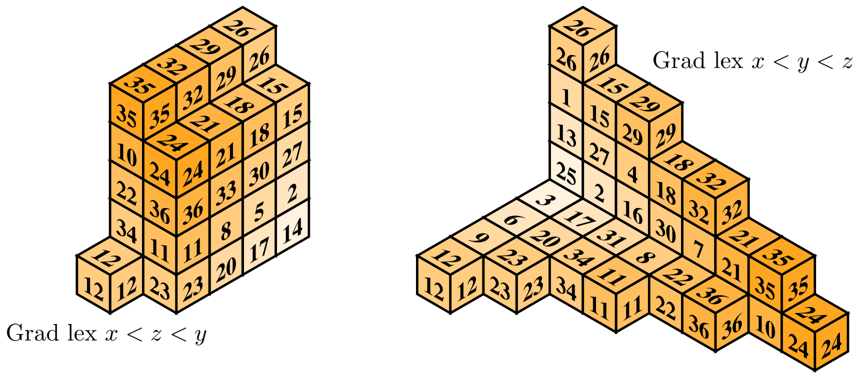


Fig. 4. Two L-shapes for $C_{37}(3, 14, 25)$

```
– $D[0, \dots, N-1] := \bar{\emptyset}$,  $S := 0$,  $\mathbf{a} := 0$
– **while** $(S < N)$
    • $c := R(\mathbf{a})$
    • **if** $D(c) = \emptyset$,
          $D(c) := \mathbf{a}$,  $S := S + 1$
    • $\mathbf{a} := s(\mathbf{a})$
```

Definition 1. *With the above notation and results, let $C_N(j_1, \dots, j_r)$ be a connected circulant digraph and $\preceq$ a graded lexicographic ordering. The monomial ideal*

$$I_C := (\mathbb{N}^r \backslash D(\mathbb{Z}_N))$$

is the ideal associated with $C_N(j_1, \dots, j_r)$ and the monomial ordering.

In the examples shown in Figure 4, we have two monomial ideals J_1 and J_2 associated with $C_{37}(3, 14, 25)$ and with respect graded lex $x < z < y$ and $x < y < z$ respectively:

$$J_1 = (x^5, x^4y, y^2, yz^4, z^5, x^4z) = (x^5, y, z) \cap (x^4, y, z^5) \cap (x^4, y^2, z^4).$$

$$J_2 = (x^5, x^4y, x^3y^2, x^2y^4, xy^6, y^8, y^7z, y^5z^2, y^3z^3, yz^4, z^5, xz) =$$
$$= (x, y^7, z^2) \cap (x, y^5, z^3) \cap (x, y^3, z^4) \cap (x, y, z^5) \cap (x^5, y, z) \cap$$
$$\cap (x^4, y^2, z) \cap (x^3, y^4, z) \cap (x^2, y^6, z) \cap (x, y^8, z).$$

Proposition 2. *With the above notations, we have*

$$\big(\mathbb{N}^r \backslash D(\mathbb{Z}_N)\big) \cap \mathbb{N}^r = \mathbb{N}^r \backslash D(\mathbb{Z}_N).$$

Obviously D is an injective map and, the cardinal of $D(\mathbb{Z}_N)$ is $N < \infty$. So, the monomial ideal I_C always contains generators of the form $X_1^{a_1}, \dots, X_r^{a_r}$, that is, the quotient polynomial ring $\mathbb{K}[X_1 \dots, X_r]/I_C$ is artinian, see Theorem 1. We say that an MDD built from a graded monomial ordenring is DEGENERATE if

I_C is an irreducible ideal, that is, if the minimal system of generators of I_C only contains such that generators. In general it is not the case as is illustrated in the above examples. The paper [10] constructed a MDD in L-shape of circulant digraphs of degree two (i.e., $r = 2$). The following concept is the generalization of L-shape to arbitrary dimension:

Definition 2. *Let I be a monomial ideal and let A be the minimal system of generators of I. We say that I is an L-shape if at most there exists one $\mathbf{x}^{\mathbf{a}} = X_1^{a_1} \cdots X_r^{a_r} \in A$ such that $a_i > 0$, for all $i = 1, \ldots, r$.*

We say that an MDD built following Algorithm 2 is an L-shape if the associated monomial ideal is an L-shape.

We will prove that any MDD built as in Algorithm 2 is an L-shape. First we need the following technical result:

Lemma 1. *With the above notation, let A be the minimal system of generators of I_C. If $\mathbf{a} \in A$ such that $a_i > 0$ for some i, then $b_i = 0$ where $D(R(\mathbf{a})) = \mathbf{b} = (b_1, \ldots, b_r)$.*

Now, we state the main result of this section:

Proposition 3. *For any circulant digraph and graded monomial ordering, Algorithm 2 returns an L-shape.*

4 Diameter and Average Distance

Two notable parameters in a digraph are the diameter and the average distance. The diameter represents the worst delay in the communication between two nodes and the average distance the average delay. In this section we show formulae to compute those parameters in a circulant digraph given by the irredundant irreducible decomposition of the monomial ideal I_C.

4.1 Diameter

Given an MDD of a circulant graph $C_N(j_1, \ldots, j_r)$ it is easy to obtain the diameter:

$$\mathrm{d}(C_N) = \max\{\|\mathbf{a}\|_1 \;/\; \mathbf{a} \in D(\mathbb{Z}_N)\}.$$

The description of the monomial ideal I_C in terms of irreducible components permits a simplification:

Proposition 4. *Let $\mathfrak{m}^{\mathbf{b}_1} \cap \cdots \cap \mathfrak{m}^{\mathbf{b}_n}$ be the irredundant irreducible decomposition of the ideal I_C. Then*

$$\mathrm{d}(C_N) = \max\{\|\mathbf{b}_i\|_1 - r \;/\; i = 1, \ldots, r\}.$$

4.2 Average Distance

Again, given an MDD of a circulant graph, it is easy to obtain the average distance:

$$\bar{d}(C_N) = \frac{\sum_{c=0}^{N-1} \|D(c)\|_1}{N} = \frac{\sum_{\mathbf{x}^{\mathbf{u}} \notin I_C} \|\mathbf{u}\|_1}{N}.$$

The following result provides a formula for computing $\bar{d}(C_N)$ for degenerate MDD.

Lemma 2. *Let $I_C = \mathfrak{m}^{\mathbf{a}+\bar{1}} = \mathfrak{m}^{\mathbf{b}}$. Then*

$$\sum_{\mathbf{x}^{\mathbf{u}} \notin I_C} \|\mathbf{u}\|_1 = \frac{b_1 \cdots b_r}{2}(b_1 + \cdots + b_r - r) = \frac{a_1 + \cdots + a_r}{2} \prod_{i=1}^{r} (a_i + 1).$$

Note 3. *In the above case, that is, when I_C is an irreducible ideal we have*

$$\bar{d}(C_N) = \frac{\mathrm{d}(C_N)}{2}.$$

To discuss the general case we will introduce some new notation. Let $\mathfrak{m}^{\mathbf{b}_1} \cap \cdots \cap \mathfrak{m}^{\mathbf{b}_n}$ be the irreducible decomposition of the monomial ideal I_C, we let

$$\mathbf{d}_\Delta := \text{exponent}\left(\gcd(\mathbf{x}^{\mathbf{b}_i} \ / \ i \in \Delta)\right), \ \forall \Delta \subseteq \{1, \ldots, n\}, \ \Delta \neq \emptyset.$$

$$\sigma(\mathbf{u}) := \frac{u_1 \cdots u_r}{2}(u_1 + \cdots + u_r - r).$$

Our next goal is to find a formula for the average distance. We will apply the general Inclusion-Exclusion Principle, as follows:

Proposition 5. *Let $I = \mathfrak{m}^{\mathbf{b}_1} \cap \cdots \cap \mathfrak{m}^{\mathbf{b}_n}$ the irreducible decomposition of the ideal I_C. We have:*

$$\sum_{\mathbf{x}^{\mathbf{u}} \notin I} \|\mathbf{u}\|_1 = \sum_{\emptyset \subsetneq \Delta \subseteq \{1,\ldots,n\}} (-1)^{\#\Delta + 1} \sigma(\mathbf{d}_\Delta).$$

Considering the ideal $I_1 = (x^4, x^2y^2, y^3)$ (see Figure 5),

$$\sum_{\mathbf{x}^{\mathbf{u}} \notin I_1} \|\mathbf{u}\|_1 = \sigma(2,3) + \sigma(4,2) - \sigma(2,2) = 9 + 16 - 4 = 21.$$

The several results introduced in Section 2 permit a strong reduction in the number of sums term that we need to consider in the expression of Proposition 5. For instance, we consider the Example 1:

$$I_2 = (x^8, y^2, z) \cap (x^7, y^2, z^4) \cap (x^4, y^3, z^2) \cap (x^4, y^5, z) \cap (x^3, y^3, z^5)$$

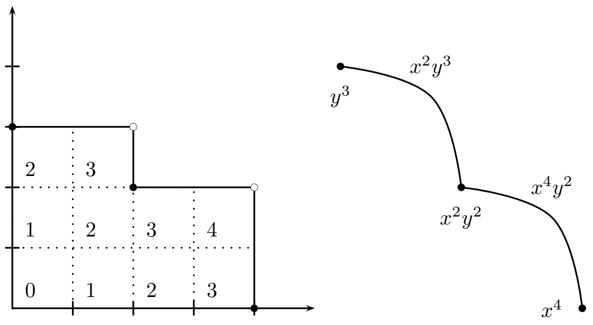


Fig. 5.

The Proposition 5 solves:

$$\begin{aligned}\sum_{\mathbf{x}^{\mathbf{u}}\notin I}\|\mathbf{u}\|_1 = \quad & \sigma(8,2,1)+\sigma(7,2,4)+\sigma(3,3,5)+\sigma(4,3,2)+\sigma(4,5,1)- \\ & -[\sigma(7,2,1)+\sigma(3,2,1)+\sigma(4,2,1)+\sigma(4,2,1)+\sigma(3,2,4)+ \\ & +\sigma(4,2,2)+\sigma(4,2,1)+\sigma(3,3,2)+\sigma(3,3,1)+\sigma(4,3,1)]+ \\ & +\sigma(3,2,1)+\sigma(4,2,1)+\sigma(4,2,1)+\sigma(3,2,1)+\sigma(3,2,1)+ \\ & +\sigma(4,2,1)+\sigma(3,2,2)+\sigma(3,2,1)+\sigma(4,2,1)+\sigma(3,3,1)- \\ & -[\sigma(3,2,1)+\sigma(3,2,1)+\sigma(4,2,1)+\sigma(3,2,1)+\sigma(3,2,1)]+ \\ & +\sigma(3,2,1)= \\ =\ & \sigma(8,2,1)+\sigma(7,2,4)+\sigma(3,3,5)+\sigma(4,3,2)+\sigma(4,5,1)+\sigma(3,2,2) \\ & -[\sigma(7,2,1)+\sigma(3,2,4)+\sigma(4,2,2)+\sigma(3,3,2)+\sigma(4,3,1)]= \\ & =454\end{aligned}$$

Clearly if $\mathbf{b}\in\mathbb{N}^r$ has a zero coordinate then $\sigma(\mathbf{b})=0$. This fact produces several cancelations in the formula of Proposition 5: $+$ for faces, $-$ for edges and $+$ for nodes.

In circulant digraphs of degree two the associated monomial ideal only has one or two irreducible components (see Proposition 3), then the computation of the average distance is immediate. For circulant digraphs of degree three we can follows this strategy (see Figure 2):

- Construct the Miller-Sturmfels' graph G as in the previous examples such that each irreducible component corresponds with the least common multiple of some generators of the minimal system of generators.
- Let E be the set of all edges, F the set of faces and N the set of vertices of G

$$\sum_{e\in F}\sigma(e)-\sum_{e\in E}\sigma(e)+\sum_{e\in N}\sigma(e).$$

5 Degenerate L-Shapes

We recall that an MDD is DEGENERATE if the associated monomial ideal is irreducible, that is, generated by monomials of the form $\left(X_1^{\alpha_1}, \ldots, X_r^{\alpha_r}\right)$. In general, graphs with this property do not need to present a low diameter. In this section we present two families of circulant graph of degree two and three having an degenerate MDD and with a relatively small diameter.

Proposition 6. *Let a, s, k natural numbers such that $\gcd(a, s) = 1$ and $a < s$. The monomial ideal associated with $C_{sk}(a, s)$ is $I_C = (x^s, y^k)$ for any monomial ordering.*

The Proposition 6 provides a family with diameter d and $O(d^2)$ nodes for each natural number $d > 1$:

$$C_{\left(\frac{d+2}{2}\right)^2}\left(1, \frac{d+2}{2}\right), \; \textit{if } d \equiv 0 \bmod 2 \quad \textit{and} \quad C_{\frac{(d+1)(d+3)}{4}}\left(1, \frac{d+1}{2}\right), \; \textit{if } d \equiv 1 \bmod 2.$$

Basically, this family was discovered in the paper [10]. However, determining $d_2(N)$ and finding the optimal $C_N(j_1, j_2)$ is an open problem. For circulant digraphs of degree three, we can prove a similar result than Proposition 6:

Proposition 7. *Let a, s, k, j be natural numbers such that $k > 1, j > 1, \gcd(a, s) = 1$ and $a < s$. The monomial ideal associated with $C_{skj}(a, s, sk)$ is $I = (x^s, y^k, z^j)$.*

For the case of undirected circulant graph of degree four, that is, $C_N(j_1, -j_1, j_2, -j_2)$ several papers showed that the lower bound $\frac{1}{2}\left(\sqrt{2N-1}-1\right)$ can be achieved by taking $j_1 = \frac{1}{2}\left(\sqrt{2N-1}-1\right)$ and $j_2 = \frac{1}{2}\left(\sqrt{2N-1}-1\right)+1$, (see the survey [2]). In the middle, that is, between circulant digraphs of degree two and circulant graphs of degree four, Proposition 7 provides a very attractive family of circulant graph of degree 3. Let $d > 2$ be a natural number:

$$C_{\left(\frac{d+3}{3}\right)^3}\left(1, \frac{d+3}{3}, \left(\frac{d+3}{3}\right)^2\right), \text{ if } d \equiv 0 \bmod 3.$$

$$C_{\frac{(d+2)^2(d+5)}{27}}\left(1, \frac{d+2}{3}, \left(\frac{d+2}{3}\right)^2\right), \text{ if } d \equiv 1 \bmod 3.$$

$$C_{\frac{(d+4)^2(d+1)}{27}}\left(1, \frac{d+4}{3}, \left(\frac{d+4}{3}\right)^2\right), \text{ if } d \equiv 2 \bmod 3.$$

They have diameter d and average distance $d/2$.

6 Conclusions

In this paper we have proposed monomial ideals as a natural tool to study circulant digraphs. We have generalized the L-shape concept in the plane to L-shape in the r-dimensional affine space. We think that this new point of view may shed a light on problems in circulant graph theory. Many interesting questions remain unsolved. Unfortunately, we do not know how to compute efficiently a minimal system of generators of the associated monomial ideal. From a more practical point of view, it should be interesting to investigate the reliabilities of the family of circulant graphs of degree three in computer networks such that routing, fault tolerance etc.

Acknowledgment. This work is partilly supported by Spanish Ministry of Science grant MTM2004-07086.

References

1. T. Beker, V. Weispfenning, *Gröbner basis - a computational approach to commutative algebra.* Graduate Texts in Mathematics. Springer Verlag, New York, Berlin, Heidelberg, 1993.
2. J.-C. Bermond, F. Comellas and D.F. Hsu. *Distributed Loop Computer Networks: A Survey.* Journal of Parallel and Distributed Computing, **24**, pp. 2-10, 1995.
3. F.T. Boesch and R. Tindell. *Circulants and their connectivity.* J. Graph Theory, **8**, pp. 487-499, 1984.
4. F. K. Hwang. *A complementary survey on double-loop networks*, Theoretical Computer Science **263**, pp. 211-229, 2001.
5. B. Mans. *Optimal Distributed algorithms in unlabeled tori and chordal rings* Journal of Parallel and Distributed Computing, **46**, pp. 80-90, 1997.
6. D.F. Hsu, Xing-De Jia, *Extremal Problems in the Combinatorial Construction of Distributed Loop Networks*, SIAM J Discrete Math, **7**, 1, pp. 57-71, 1994.
7. E. Miller, *Resolutions and Duality for Monomial Ideals.* Ph.D. thesis, 2000.
8. E. Miller, B. Sturmfels, *Monomial Ideal and Planar Graphs.* Proceedings of AAECC-13 (Honolulu, Nov. 1999), Springer LNCS 1719, pp. 19-28.
9. B. Sturmfels, *Gröbner Bases and Convex Polytopes*, University Lecture Series, **8** American Mathematical Society, 1996.
10. C.K. Wong, D. Coppersmith, *A Combinatorial Problem Related to Multimodule Memory Organizations*, J. ACM **21**, 3, pp. 392-402, 1974.
11. J. Žerovnik, T. Pisanski, *Computing the Diameter in Multiple-Loop Networks.* J. Algorithms **14**, 2, pp. 226-243, 1993.

Algebraic Topological Analysis of Time-Sequence of Digital Images[⋆]

Rocio Gonzalez–Diaz, Belen Medrano[⋆⋆], Pedro Real, and Javier Sánchez–Peláez

Depto. Matematica Aplicada I, E.T.S.I. Informatica, Universidad de Sevilla, Avda. Reina Mercedes, s/n 41012 Sevilla, Spain
{rogodi, belenmg, real}@us.es
http://www.us.es/gtocoma

Abstract. This paper introduces an algebraic framework for a topological analysis of time-varying $2D$ digital binary–valued images, each of them defined as 2D arrays of pixels. Our answer is based on an algebraic-topological coding, called AT–model, for a nD ($n = 2, 3$) digital binary-valued image I consisting simply in taking I together with an algebraic object depending on it. Considering AT–models for all the 2D digital images in a time sequence, it is possible to get an AT–model for the 3D digital image consisting in concatenating the successive 2D digital images in the sequence. If the frames are represented in a quadtree format, a similar positive result can be derived.

1 Introduction

In [6,7], a method for computing cohomology aspects of three–dimensional digital binary-valued images is described. That work is mainly based on two facts: (1)to consider a simplicial model $K(I)$ for a digital image I using a $(14, 14)$–adjacency relation between voxels; and (2)to apply an "algebraic homological process" in which an algebraic artifact c (a special type of chain homotopy equivalence [11]) connecting the chain complex canonically associated to the simplicial version of the digital image with its homology is constructed. An AT-model (algebraic-topological model) for the $3D$ digital image I is the couple (I, c). Roughly speaking, an AT-model is an extra algebraic-topological information of the image. This particular description for digital images used there for solving a problem of topological interrogation, is used in this paper for solving a problem of topological analysis. We are interested here in understanding the topological nature of a time-sequence of $2D$ digital binary-valued images. There are two ways for handling this question: (1) [the intraframe approach] to determine the "topology" of each frame and to try to adequately join these pieces in order to give a correct three–dimensional topological interpretation, or (2) [the $3D$ approach] to

⋆ This work has been partially supported by the PAICYT research project FQM–296 from Junta de Andalucia.

⋆⋆ Fellow associated to University of Seville under a Junta de Andalucia research grant.

directly obtain the topological information from the $3D$ image consisting in concatenating the successive $2D$ images of the sequence. Using AT-models, we will see here that both strategies lead us to the same result. This is also valid in the case in which the $2D$-images are represented under a quadtree format. To extend these positive results to time-sequences of $3D$ digital binary–valued images is an extremely interesting challenge which seems to be affordable.

2 Digital Images and Simplicial Representations

In this paper, a nD $(n = 2, 3)$ digital binary-valued image is a nD array of elements (pixels in $2D$, voxels in $3D$) whose positions are represented by integer coordinates and whose values can only be 1 (black) or 0 (white). Given a pixel (x_1, x_2), its 6-adjacent pixels are $(x_1-1, x_2), (x_1+1, x_2), (x_1, x_2+1), (x_1+1, x_2+1), (x_1, x_2-1), (x_1-1, x_2-1)$. It is immediate to define another 6-connectivity if we favour the direction 135^o for determining adjacent pixels instead of 45^o. The 6-connectivity in $2D$ digital images satisfies the Jordan curve property. This ensures that a black simple closed curve (that is, a set of black pixels C such that each pixel in C have exactly two 6-neighbours in C) will separate the background into two non-6-adjacent white regions, the interior and the exterior. Given a voxel (x_1, x_2, x_3), its 14-neighbour voxels are showed in Figure 1. It is possible to define other types of 14-adjacency, favouring other directions. The 14-adjacency in $3D$ defined by the neighbour relations given in Figure 1, is an appropriate generalization of the 6-adjacency in 2D previously defined.

In [7], the 14-adjacency relation is used in order to be able to naturally associate a three-dimensional simplicial complex $K(I)$ to any $3D$ digital binary–valued image I. $K(I)$ is called a *simplicial representation of* I and is defined as a subcomplex of the infinite simplicial complex K obtained by the decomposition of the 3D euclidean space into unit cubes (which vertices are the points $(a, b, c) \in \mathbf{Z}^3$), and the decomposition of each cube into six tetrahedra as shown in Figure 2 (the six tetrahedra are: $\langle 1, 3, 4, 8\rangle$, $\langle 1, 2, 4, 8\rangle$, $\langle 1, 2, 6, 8\rangle$, $\langle 1, 3, 7, 8\rangle$, $\langle 1, 5, 7, 8\rangle$, $\langle 1, 5, 6, 8\rangle$.) Two digital images, I_1 and I_2, are isomorphic if and only if their simplicial representations $K(I_1)$ and $K(I_2)$ are homeomorphic.

In order to give a formal definition of simplicial representation, we need to give some preliminaries. Our terminology follows [12]. Considering an ordered

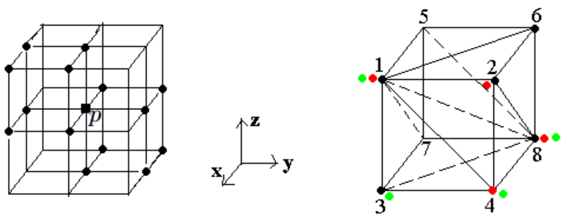


Fig. 1. The 14–neighbours of a voxel p (on the left) and the decomposition of a unit cube into six tetrahedra (on the right)

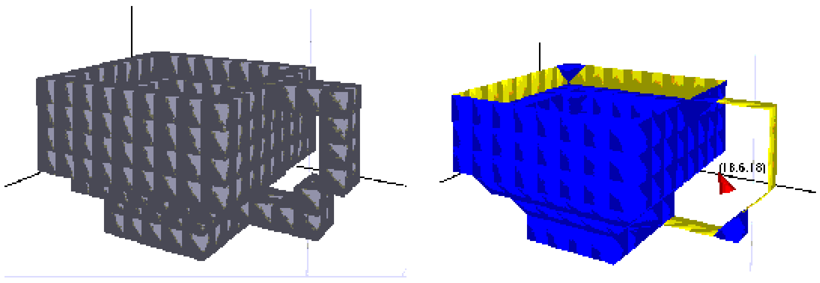


Fig. 2. A digital binary image and its simplicial representation using 14-adjacency relation between voxels

vertex set V, a q–simplex with $q+1$ affinely independent vertices $v_0 < \cdots < v_q$ of V is the convex hull of these points, denoted by $\langle v_0, \ldots, v_q \rangle$. If $i < q$, an *i–face* of σ is an i–simplex whose vertices are in the set $\{v_0, \ldots, v_q\}$. A *simplicial complex* K is a collection of simplices such that every face of a simplex of K is in K and the intersection of any two simplices of K is a face of each of them or empty. The set of all the q–simplices of K is denoted by $K^{(q)}$.

Let K and L be simplicial complexes and let $|K|$ and $|L|$ be the subsets of $\mathbf{R}^d$ that are the union of simplices of K and L, that is, the geometric realizations of K and L, respectively. We say that K and L are homotopic if its respective geometric realization are homotopy equivalents. It is known that if two spaces are homeomorphic then they are homotopy equivalent.

Given a nD $(n = 2, 3)$ digital binary–valued image I and considering the lexicographical ordering on $\mathbf{Z}^3$, a *simplicial representation* $K(I)$ of I is the simplicial complex described as follows: the i–simplices of $K(I)$ $(i \in \{0, 1, 2, 3\})$ are constituted by the different sorted sets of $(i+1)$ 14–neighbour black voxels of I. Moreover, let I be a $2D$ digital binary–valued image embedded in the 3D digital space. The simplicial representation $K(I)$ of I is the two-dimensional simplicial complex whose i–simplices $(i \in \{0, 1, 2\})$ are constituted by the different sorted sets of $(i+1)$ 6–neighbour black pixels of I.

3 An AT-Model for a Digital Image

In this section we briefly recall the notion of AT-model for digital images given in [6,7]. For explain it, we first define the concept of chain contraction that is an exotic notion in the field of Digital Topology but it is a common resource in Algebraic Topology (see, for example, [11]).

Since the objects considered in this paper are embedded in $\mathbf{R}^3$ then the homology groups vanish for dimensions greater than 3 and they are torsion–free for dimensions 0, 1 and 2 (see [1–ch.10]). Therefore, for simplicity, we can consider that the ground ring is $\mathbf{Z}/\mathbf{Z}2$ throughout the paper. Nevertheless, all the procedure we explain here, is valid for any commutative ring.

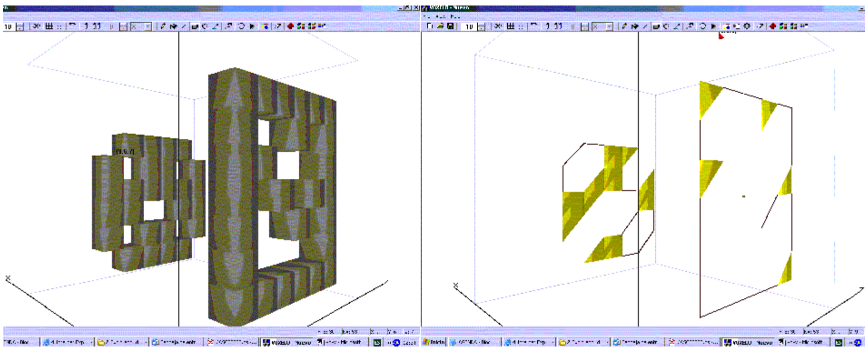


Fig. 3. Two $2D$ images embedded in the $3D$ digital space and its respective simplicial representations

Let K be a simplicial complex. A *q–chain a* is a formal sum of simplices of $K^{(q)}$. We denote $\sigma \in a$ if $\sigma \in K^{(q)}$ is a summand of a. The q–chains form a group with respect to the component–wise addition; this group is the *qth chain group* of K, denoted by $C_q(K)$. There is a chain group for every integer $q \geq 0$, but for a complex in $\mathbf{R}^3$, only the ones for $0 \leq q \leq 3$ may be non–trivial. The *boundary* of a q–simplex $\sigma = \langle v_0, \ldots, v_q\rangle$ is the collection of all its $(q-1)$–faces which is a $(q-1)$–chain: $\partial_q(\sigma) = \sum\langle v_0, \ldots, \hat{v}_i, \ldots, v_q\rangle$, where the hat means that v_i is omitted. By linearity, the boundary operator ∂_q can be extended to q–chains. The collection of boundary operators connect the chain groups $C_q(K)$ into the *chain complex* $C(K)$ canonically associated to K: $\cdots \overset{\partial_2}{\to} C_1(K) \overset{\partial_1}{\to} C_0(K) \overset{\partial_0}{\to} 0$.

In a more general framework, a chain complex $\mathcal{C}$ is a sequence $\cdots \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} 0$ of abelian groups C_q and homomorphisms d_q, indexed with the non–negative integers, such that for all q, $d_q d_{q+1} = 0$. A chain $a \in C_q$ is called a *q–cycle* if $d_q(a) = 0$. If $a = d_{q+1}(a')$ for some $a' \in C_{q+1}$ then a is called a *q–boundary*. Define the *qth homology group* to be the quotient group of q–cycles and q–boundaries, denoted by $H_q(\mathcal{C})$. Let $\mathcal{C} = \{C_q, d_q\}$ and $\mathcal{C}' = \{C'_q, d'_q\}$ be two chain complexes. A *chain map* $f : \mathcal{C} \to \mathcal{C}'$ is a family of homomorphisms $\{f_q : C_q \to C'_q\}_{q\geq 0}$ such that $d'_q f_q = f_{q-1} d_q$.

Definition 1. *[11] A* chain contraction *of a chain complex $\mathcal{C}$ to another chain complex $\mathcal{C}'$ is a set of three homomorphisms $c = (f, g, \phi)$ such that: (i) $f : \mathcal{C} \to \mathcal{C}'$ (called projection) and $g : \mathcal{C}' \to \mathcal{C}$ (called inclusion) are chain maps. (ii) fg is the identity map of $\mathcal{C}'$. (iii) $\phi : \mathcal{C} \to \mathcal{C}$ is a chain homotopy of degree +1 of the identity map $id_{\mathcal{C}}$ of $\mathcal{C}$ to gf, that is, $id_{\mathcal{C}} + gf = \phi d + d\phi$, where d is the boundary operator of $\mathcal{C}$.*

Important properties of chain contractions are: (a) $\mathcal{C}'$ has fewer or the same number of generators than $\mathcal{C}$; (b) $\mathcal{C}$ and $\mathcal{C}'$ have isomorphic homology groups [12–p. 73].

Now, we are ready to define an AT-model for a digital image.

Definition 2. *[7] An* algebraic topological model *(more briefly called AT-model) for a nD ($n = 2, 3$) digital image I is the couple $M_I = (I, (f_I, g_I, \phi_I))$ where $c_I = (f_I, g_I, \phi_I)$ is a chain contraction of the chain complex $C(K)$ to a chain complex $\mathcal{H}$,being K a simplicial complex homotopic to the simplicial representation $K(I)$ and $\mathcal{H}$ a chain complex isomorphic to the homology of I.*

It is necessary to emphasize that an AT- model is non–unique. First, K can be any simplicial complex homotopic to the simplicial representation $K(I)$ and $\mathcal{H}$ any chain complex isomorphic to the homology of I. Second, the morphisms f_I, g_I and ϕ_I can admit different formulas, all of them allowing to define different c_I of $C(K)$ to $\mathcal{H}$.

Proposition 1. *Let I be a nD ($n = 2, 3$) digital binary–valued image. There is an algorithm calculating an AT-model for I.*

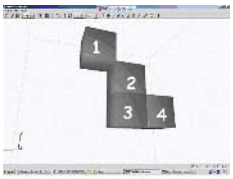
We construct the desired chain contraction adequately modifying the classical algorithms for computing homology (matrix, incremental,...) existing in the literature. For example, the matrix algorithm [12] is based on the reduction of the matrices defining the boundary operator to Smith normal form, from which one can read off the homology groups of the complex. A chain homotopy equivalence version of this process is given in [4]. An algebraic homological output for an incremental algorithm [3] for computing homology is given in [6]. The complexity of both algorithms computing AT-models is $O(r^3)$, where r is the number of black pictures elements (pixels or voxels).

Example 1. The algebraic–topological model of the image J showed in Figure 4 is:

	$\langle 1\rangle$	$\langle 2\rangle$	$\langle 3\rangle$	$\langle 4\rangle$	$\langle 1,2\rangle$	$\langle 2,3\rangle$	$\langle 2,4\rangle$	$\langle 3,4\rangle$	$\langle 2,3,4\rangle$
f	$\langle 1\rangle$	$\langle 1\rangle$	$\langle 1\rangle$	$\langle 1\rangle$	0	0	0	0	0
g	$\langle 1\rangle$								
ϕ	0	$\langle 1,2\rangle$	$\langle 2,3\rangle + \langle 1,2\rangle$	$\langle 2,4\rangle + \langle 1,2\rangle$	0	0	0	$\langle 2,3,4\rangle$	0

$\mathcal{H}$ has only one generator in dimension 0 represented by the simplex $\langle 1\rangle$. Since $\mathcal{H}$ is isomorphic to $H(J)$ we get that $H(J)$ has only one generator, too.

Let us suppose that we have computed an AT-model $M_I = (I, (f_I, g_I, \phi_I)$ for a digital image I. We are interested here in a full understanding of the structures and morphisms involved in M_I. Interesting properties of the morphisms f_I, g_I



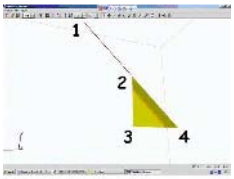


Fig. 4. The black points of the image J and its simplicial representation

and ϕ_I are that for all $x \in C(K)$ and $z \in \mathcal{H}$: (1) $f_I\partial(x) = 0$; (2) $\partial g_I(z) = 0$; In fact, $g_I(z)$ is a representative cycle of the homology generator z; (3) $\partial(x) = \partial\phi_I\partial(x)$; (4) If $\partial(x) = 0$, then $x + g_If_I(x) = \partial\phi_I(x)$; (5) If $x = \partial(y)$, then $x = \partial\phi_I(x)$.

It is possible to simplify the definition of an AT-model for a nD digital image.

Proposition 2. *An* AT-model *for a nD ($n = 2, 3$) digital image I can be represented by a couple $M_I = (I, \phi_I)$ where $\phi_I : K \to C(K)$ is a linear map of degree 1 such that $\phi\phi = 0$, $\phi\partial\phi = \phi$ and $\partial\phi\partial = \partial$, where ∂ is the boundary operator in $C(K(I)$ and K is a simplicial complex homotopic to the simplicial representation $K(I)$.*

Proof. The proof of the previous result is mainly based on two well-known facts. First, given a chain contraction $c = (f, g, \phi)$ from $\mathcal{C}$ to $\mathcal{C}'$, it is possible to construct another contraction $c' = (f, g, \phi')$ from $\mathcal{C}$ to $\mathcal{C}'$ such that the chain homotopy ϕ' satisfies the following additional conditions: $(iv)\phi'\phi' = 0$; $(v)\phi' g = 0$; $(vi)f\phi' = 0$. In fact, the formula for ϕ' is

$$\phi' = (\partial\phi + \phi\partial)\phi(\partial\phi + \phi\partial)\partial(\partial\phi + \phi\partial)\phi(\partial\phi + \phi\partial),$$

being ∂ the boundary operator of $\mathcal{C}$. Second, a chain contraction $c = (f, g, \phi)$ from $\mathcal{C}$ to $\mathcal{C}'$ satisfying $(i) - (vi)$ conditions is equivalent to give a map $\phi' : \mathcal{C} \to \mathcal{C}$ (called splitting homotopy) satisfying the following conditions: (1) ϕ' is a linear map of degree $+1$; (2) $\phi'\phi' = 0$; and (3) $\phi'\partial\phi' = \phi'$, being ∂ the boundary operator of $\mathcal{C}$. Let $c = (f, g, \phi)$ be a chain contraction $c = (f, g, \phi)$ from $\mathcal{C}$ to $\mathcal{C}'$ satisfying $(i) - (vi)$ conditions. Applying ϕ to (iii) and using the other identities shows that $\phi\partial\phi = \phi$. Then, the desired ϕ' is ϕ. Conversely, let $\phi' : \mathcal{C} \to \mathcal{C}$ be a map satisfying $(1) - (3)$. Let $\pi = id_{\mathcal{C}} - \partial\phi' - \phi'\partial$. Then, $\mathcal{C} = im(\pi) \oplus ker(\pi)$. We define the chain contraction $c' = (f', g', \phi')$ from $\mathcal{C}$ to $im(\pi)$, where f' is the corestriction of π and g' is the inclusion. Then, using the equality $\pi^2 = \pi$ and the condition (3), it is easy to prove that $c' = (f', g', \phi')$ is the desired chain contraction.

Now, we are going to prove the proposition. Let $M_I = (I, (f, g, \phi))$ be an AT-model for a nD digital image. Then, $c = (f, g, \phi)$ is a chain contraction from $C(K)$ to $\mathcal{H}$, being $\mathcal{H}$ a chain complex isomorphic to the homology of I. Using the result previous, we can suppose that the contraction satisfies $(i) - (vi)$ and then we have a splitting homotopy ϕ' from $C(K)$ to $C(K)$ satisfying $(1) - (3)$. From ϕ', we have the new chain contraction $c' = (f', g', \phi')$ from $C(K)$ to $im(\pi)$. Let us now show that $im(\pi)$ has null boundary operator. If $x = \pi(y) \in im(\pi)$, then $\partial(x) = \partial\pi(y) = \partial g'f'(y) = g'\partial f'(y) = 0$ since $\mathcal{H}$ is a chain complex isomorphic to the homology of I. Using this last result and applying the operator ∂ to condition (iii) of c', we get the equality $\partial = \partial\partial\phi' + \partial\phi'\partial$. Since $\partial\partial = 0$, we finally obtain that $\partial\phi'\partial = \partial$. Conversely, let $\phi : C(K) \to C(K)$ be a linear map of degree 1 such that $\phi\phi = 0$, $\phi\partial\phi = \phi$ and $\partial\phi\partial = \partial$. We now construct the chain contraction $c = (f, g, \phi) : C(K) \to im(\pi)$. We are going to show that c defines a AT-model for the digital image I. Let d be the boundary operator of

$im(\pi)$. We have to prove that $d = 0$. Applying ∂ to condition (iii) of c and using that $\partial\partial = 0$ and $\partial\phi\partial = \partial$, we obtain $\partial - \partial gf = \partial$. Therefore, $\partial gf = gdf = 0$. Since f is onto and g is one-to-one, we show that $d = 0$.

Briefly, the additional algebraic-topological information showed in a AT-model for a nD $(n = 2, 3)$ digital binary–valued image I can be then codified in terms of a chain homotopy satisfying certain conditions. From that algebraic germ, one can form a chain contraction determining the homology of the simplicial complex $K(I)$.

4 Determining AT-Models Using Other Representation Schemes of Digital Images

It is possible to compute an AT-model for a digital image using other representation schemes. As an example, let us see how to compute this model using the quadtree representation of a $2D$ digital binary–valued image.

A *quadtree representation* (see, for example, [14]) of an image I is a tree whose leaves represent quadrants of the image and are labelled with the color of the corresponding area, i.e, black or white. In order to obtain the quadtree representation of a digital image, first the whole image is decomposed into four equal–sized quadrants. If one of the quadrants does not contain a uniform region (black or white), it is again subdivided into four quadrants. The decomposition stops if only uniform quadrants are encountered. The recursive decomposition is then represented in a data structure known as tree. Each leaf node of a quadtree representation can be assigned a unique locational code corresponding to a sequence of directional codes that locate the leaf node along a path starting at the root of the tree. A black node of a digital image is encoded as $A = (A_1 A_2 \dots A_n)$ with digits in the set $S = \{1, 2, 3, 4\}$ for A_i, where each digit in the sequence represents the quadrant subdivision from which it originates (see Figure 5). The quadrant is defined as the collection of all black node descriptions.

In order to construct the simplicial complex $K(Q(I))$ associated to the quadtree representation $Q(I)$ of the $2D$ digital image, we need first to find the neighbours of each leaf node of the quadtree $Q(I)$.

Node B is said to be a *neighbour* of node A in direction D if B corresponds to the block adjacent to A in direction D of size equal, larger or smaller than the block corresponding to A. Hence a node can have no neighbour, one or more neighbours in a chosen direction. Using a method similar to that given in [15] and considering 6-adjacency in a $2D$ digital image I, it is a simple exercise to compute the neighbours of a leaf of the quadtree representation $Q(I)$.

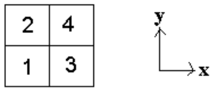


Fig. 5. The quadrant subdivision

Now, starting from the quadtree representation $Q(I)$ of a digital image I, we construct a simplicial complex $K(Q(I))$ as follows: the vertices (or 0–simplices) of $K(Q(I))$ are the leaves of $Q(I)$. The i–simplices of $K(Q(I))$ ($i \in \{1, 2\}$) are constituted by the different sorted sets of $i + 1$ neighbour leaves of $Q(I)$.

Now, it is immediate to see that the geometric realization $|K(Q(I))|$ can be obtained from $|K(I)|$ continuously "deforming" black nodes of I into points.

Proposition 3. *The simplicial representation $K(I)$ of a digital image I is homotopic to the simplicial representation $K(Q(I))$ of the quadtree representation $Q(I)$ of I.*

Since $K(Q(I))$ have, in general, much less number of simplices than $K(I)$, the computation of an algebraic topological model for I using $K(Q(I))$ may be much faster that using $K(I)$.

5 The Topological Complexity of a Time-Sequence of 2*D* Digital Images

We show here that the AT-model technique is well fitted to the problem of analysing the topology of a time-sequence of $2D$ digital images. Let us recall that there are two ways for handling this question: the intraframe and the $3D$ approaches.

Let $(I_1, I_2, \ldots, I_s)$ be a time-sequence of $2D$ digital binary–valued images. Let V_r, with $1 \leq r \leq s$, the $3D$ digital image resulting of concatenating the successive $2D$ images $I_1, I_2, \ldots, I_r$. This fact is noted by $V_r = I_1 + I_2 + \ldots + I_r$. Let us define *an AT-model for a time-sequence of* $2D$ *digital images* as an AT-model for the volume V_s. One method is to directly apply a known algorithm

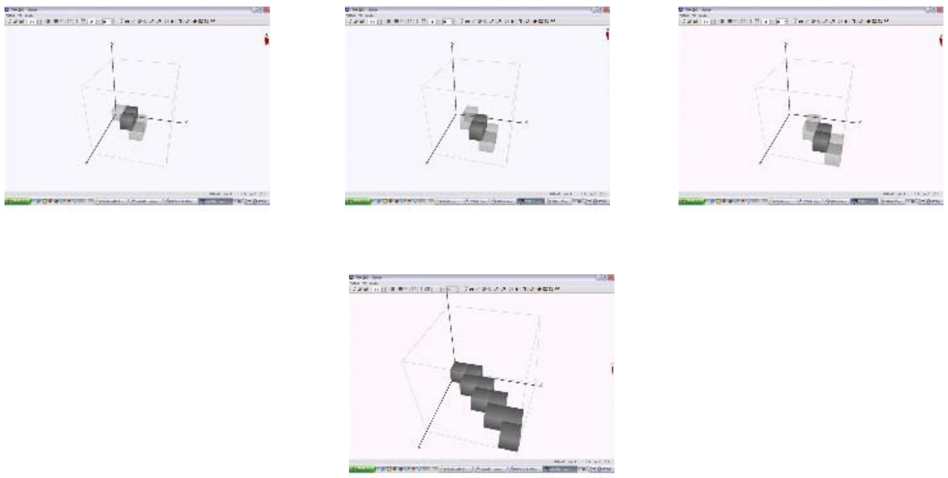


Fig. 6. Several frames of a simple time-sequence and the associated 3D binary image

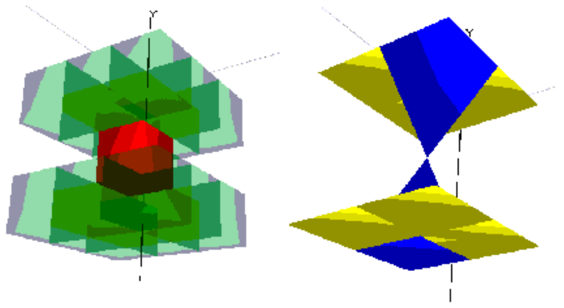


Fig. 7.

for computing an AT-model for the simplicial representation of V_s. On the other hand, starting from the respective AT-models $\{M_{I_1}, M_{I_2}, \dots, M_{I_s}\}$ for all the frames in the sequence, it is possible to adequately "gluing" them in order to form an AT-model for the "concatenated" $3D$ image V_s. Let us suppose that we have constructed an AT-model M_{V_r} for the $3D$ image V_r (with $r < s$). Starting from M_{V_r} and $M_{I_{r+1}}$, it is possible to generate an AT-model for V_{r+1}. The chain complex $C(K(V_r \cup K(I_{r+1})))$ coincides with the direct sum of the chain complexes $C(K(V_r)) \oplus C(K(I_{r+1}))$. It is immediate to construct the chain contraction $c_{V_r} \oplus c_{I_{r+1}}$ from $C(K(V_r + I_{r+1}))$ to $H(K(V_r))) \oplus H(K(I_{r+1})))$. Let f^r, g^r and ϕ^r be the morphism of that chain contraction and let h^r be the set of all the representative homology generators of $H(K(V_r) \oplus H(K(I_{r+1})$. Let $\{\tau_1, \dots, \tau_\ell\}$ be the simplices "connecting" the simplicial complexes $K(V_r)$ and $K(I_{r+1})$ such that $\{\text{simplices of } K(V_r)\} \cup \{\text{simplices of} K(I_{r+1}))\} \cup \{\tau_1, \dots, \tau_r\}$, with $r \leq \ell$ is a subcomplex of $K(V_{r+1}))$. These simplices are perfectly determined for each voxel in frame $r+1$. As an example, it is showed to the left of Figure 7, the frame 5 (consisting in one voxel in red) and frames 4 and 6 (consisting respectively in nine voxels in green) of a sequence and to the right we have the simplicial representation of these frames using 14-adjacency relations.

Now, let us use an incremental algorithm for computing an AT-model for a time-sequence of $2D$ digital binary-valued images knowing AT-models for the frames.

```
Algorithm 1. INPUT: The sorted set of simplices $\{\tau_1, \dots, \tau_\ell\}$ of $K(V_{r+1})$
OUTPUT: a splitting homotopy defining an AT-model for the volume $V_{r+1}$
Initially, $f_{alg}(\sigma) := f^r(\sigma)$, $\phi_{alg}(\sigma) := \phi^r(\sigma)$ $\forall \sigma \in K(V_r) \cup K(I_{r+1})$; and $h := h^r$
For $i = 1$ to $i = \ell$ do
    If $f_{alg}\partial(\sigma_i) = 0$ then
        $h := h \cup \{\sigma_i\}$
        $f_{alg}(\sigma_i) := \sigma_i$.
    Else take any one $\sigma_j$ of $f_{alg}\partial(\sigma_i)$, then
        $h := h - \{\sigma_j\}$,
        For $k = 1$ to $k = m$ do
            If $\sigma_j$ appears in $f_{alg}(\sigma_k)$ then
```

```
                $f_{alg}(\sigma_k) := f_{alg}(\sigma_k) + f_{alg}\partial(\sigma_i)$
                $\phi_{alg}(\sigma_k) := \phi_{alg}(\sigma_k) + \sigma_i + \phi_{alg}\partial(\sigma_i)$
            End if.
        End for.
    End if.
End for.
OUTPUT: the splitting homotopy $\phi'$ obtained from the homotopy $\phi_{alg}$.
```

Therefore, we have

Theorem 1. *Given a time-sequence $I_1, \ldots, I_s$ of $2D$ digital binary–valued images represented as $2D$ pixel arrays, it is possible to compute an AT-model for the sequence from the information given by the AT-models for all the frames.*

If we deal with quadtree representations for the $2D$ digital images of the sequence, the AT-model technique works well. The key idea is to understand how the simplicial representation $K(V_s)$ based on 14-adjacency relations between voxels of V_s is deformed in such a way that each frame I_j $(j = 1, \ldots, s)$ is represented now by the quadtree simplicial representation $K(Q(I_j))$. The resulting simplicial complex of this process is denoted by $K_Q(V_s)$. Determining that this simplicial complex is homotopic to the simplicial representation $K(V_s)$ is an elementary question. The intraframe approach here gives the desired AT-model for the volume V_s. Let us start from AT-models for all quadtree frames in the sequence and take into account that neighbour nodes in frame $j+1$ of a node in frame j are perfectly determined by the 14-adjacency relations. Then, an similar algorithm to the previous one can be applied to this situation.

Theorem 2. *Given a sequence $I_1, \ldots, I_s$ of $2D$ digital binary–valued images represented under quadtree format, it is possible to compute an AT-model for the sequence from the information given by the quadtree AT-models for all frames.*

6 Conclusions

In this paper, we are interested in providing an algebraic solution to the problem of topologically analysing a sequence of 2D digital binary-valued images The method is based on the notion of AT-model for a nD $(n = 2, 3)$ digital image and both the intraframe and $3D$ approaches give rise to the same positive results. It seems possible to extend this method to sequences of $3D$ digital binary–valued images taking into consideration a 30-adjacency relation between tetraxels (elemental picture elements of dimension 4). In positive case, another interesting question is if the octree representation for $3D$ digital images could be successfully used in our algebraic-topological setting. On the other hand, an AT-model for a $4D$ digital image would allow the computation of highly abstract algebraic invariants such as cohomology operations [5] in an efficient way. These new computational tools in the setting of Digital Imaging may be used to facilitate the topological classification problem in $4D$.

It is possible to take advantage of temporal coherence of time-varying 2D digital binary-valued images (coherence between consequent time frames) to obtain a more efficient version of Algorithm 1. Having a time-sequence $\{I_1, \ldots, I_s\}$, the idea is to take a "differential" coding of this sequence as, for example, $\{I_1, D_2, D_3, \ldots, D_s\}$, where $D_i = I_{i-1} \text{xor} I_i$, for all $i = 2, \ldots, s$. From an AT-model for the first frame I_1, it is possible to gradually generate AT-models for I_r and V_r ($r = 2, \ldots, s$) by means of a similar method to Algorithm 1. An interesting question would be to determine the complexity of this process.

The results of the previous section show us in particular that an AT-model for a digital image is essentially a reusable datum. In general, from AT-models for simple images, it seems to be possible to directly "manipulate" them in order to obtain an AT-model for a more complex image. To confirm this intuition would be an important result for this theory.

The idea of constructing an "continuous analog" (a polyhedron in this case) reflecting the topology of the digital image as Euclidean subspace goes back to the origin of Digital Topology (see [13,9] for an introduction to the topics in this area). In order to develop a mathematical theory with high computational and combinatorial flavour, a fundamental goal in this area has been to try to derive purely combinatorial algorithms from the previous algebraic topological scenario. Our method is based on a pure algebraic notion (a chain homotopy equivalence) which is fundamental for both describing the topological complexity of a digital image and enriching the list of digital topological invariants. Further research must be made in order to design an appropriate "digital topology theory" taking as main notion a combinatorial version of a chain homotopy equivalence.

Finally, the application of algebraic topology-based machinery to the field of digital images could be of interest in problems of topological control in Solid Modeling and Computer Aided Geometric Design. An interesting challenge is to know if the AT-model is well adapted to control the topological complexity of a digital image transformed by a digital (local or global) operation.

References

1. Alexandroff P., Hopf H.: Topologie I. Springer, Berlin 1935
2. Barnes D. and Lambe L.: A fixed approach to homological perturbation theory. Proc. Amer. Math. Soc. **112** (1991) 881–892.
3. Delfinado C.J.A., Edelsbrunner H.: An Incremental Algorithm for Betti Numbers of Simplicial Complexes on the 3–Sphere. Comput. Aided Geom. Design **12** (1995) 771–784
4. Computation of Cohomology Operations on Finite Simplicial Complexes. Homology, Homotopy and Applications, vol 5(2) (2003) 83-93.
5. González–Díaz R., Real P.: Geometric Object and Cohomology Operations, Proceeding CASC 2002, 121–130.
6. González-Díaz R., Real P.: Towards Digital Cohomology. Lecture Notes in Computer Science **vol. 2886** (2003) 92-101.
7. González–Díaz R., Real P.: On the Cohomology of 3D Digital Images. Discrete Applied Math **vol. 147**, Issues 2–3 (2005) 245–263.

8. González–Díaz R., Medrano B., Real P., Sánchez–Peláez J.: Topological control in digital images. In preparation.
9. Kong T.Y., Roscoe A.W., Rosenfeld A.: Concepts of Digital Topology. Topology and its Applications **46** (1992) 219–262
10. Lambe L., Stasheff J.: Applications of Perturbation Theory to Iterated Fibrations. Manuscripta Math. **58** (1987) 363–376.
11. MacLane S.: Homology. Classic in Math., Springer–Verlag (1995)
12. Munkres J.R.: Elements of Algebraic Topology. Addison–Wesley Co. 1984
13. Rosenfeld A.: 3D Digital Topology. Inform. and Control **50** (1981) 119–127.
14. Rosenfeld A., Kak A.C.: Digital Picture Processing, vol 2. Academic Press, 1982.
15. Vöros J.: A strategy for repetitive neighbor finding in images represented by quadtrees. Pattern Recognition Letters **18** (1997) 955–962.

Hilbert Stratification and Parametric Gröbner Bases

Laureano Gonzalez–Vega[1], Carlo Traverso[2], and Alberto Zanoni[2]

[1] Dpto. Matemáticas, Estadística y Computación,
Facultad de Sciencias, Universidad de Cantabria,
Avenida de Los Castros s/n - E-39071 Santander, Spain
gvega@matesco.unican.es
[2] Dipartimento di Matematica "Leonida Tonelli", Università di Pisa,
Via F. Buonarroti 2 – 56127 Pisa, Italy
{traverso, zanoni}@posso.dm.unipi.it

Abstract. In this paper we generalize a method to analyze inhomogeneous polynomial systems containing parameters. In particular, the Hilbert function is used as a tool to check that the specialization of a "generic" Gröbner basis of the parametric polynomial system (computed in a polynomial ring having both parameters and unknowns as variables) is a Gröbner basis of the specialized system. Extending the analysis, we can also build the so-called *Hilbert stratification* of the associated variety. We classify the possible specializations according to the value of the Hilbert function of the specialized system. Some computation examples with the `PoSSoLib` are reported.

Keywords and phrases: Gröbner bases, Hilbert function, Specialization.

AMS Subject Classification: 68W30, 13P10, 13F20, 13D40.

1 Introduction

Let $\mathbf{T} = (T_1, \ldots, T_m)$ be the parameters, $\mathbf{X} = (X_1, \ldots, X_l)$ be the unknowns and $F = \{f_1, ..., f_r\}$ be a set of polynomials in $\mathbb{K}[\mathbf{T}, \mathbf{X}]$, where $\mathbb{K}$ is a field. The problem to solve is the determination (whatever it means — it is clarified below) of the solution set of the polynomial system of equations:

$$F = F(\mathbf{T}, \mathbf{X}) = \{f_1(\mathbf{T}, \mathbf{X}) = \cdots = f_r(\mathbf{T}, \mathbf{X}) = 0\}$$

Two different questions arise naturally in this context: first, to get the conditions the parameters must satisfy for the considered system to have a solution and, second, to describe in some way the solutions, i.e. the dependency between every unknown and the parameters.

Various methods are present in the Computer Algebra literature for this general problem: in the following works one can find some necessary background.

- Comprehensive Gröbner bases [17,18].
- Dynamic Evaluation Method [4].
- Multivariate resultants [16].
- Triangular sets (see [13] and references therein).
- DISPGB algorithm [12].
- Specific Linear Algebra tools for parametric linear systems [14].

The method of comprehensive Gröbner bases is the most general and powerful one, but is also the most difficult to apply in its full generality, because of its great complexity. The other ones are limited in their applications, but more practical and feasible, and the present work tries to follow this philosophy. We'll describe a procedure to obtain some semialgebraic sets in the parameter space, whose union gives a "good" set of sufficient condition for specialization.

2 Gröbner Bases and Specializations

Let $\mathbb{K} \subseteq \widetilde{\mathbb{K}}$ be fields, $\mathbf{X} = (X_1, \ldots, X_l)$, $\mathbf{T} = (T_1, \ldots, T_m)$ be sets of variables.

Definition 1. *A* specialization *is an algebra homomorphism* $\varphi : \mathbb{K}[\mathbf{T}] \to \widetilde{\mathbb{K}}$ *The* specialization associated with a point $t \in \widetilde{\mathbb{K}}^m$ *is defined by* $\varphi_t(p(\mathbf{T})) := p(t)$*, for* $p(\mathbf{T}) \in \mathbb{K}[\mathbf{T}] \subseteq \widetilde{\mathbb{K}}[\mathbf{T}]$*.*

In the paper we'll use the following variant, which fits our needs.

Definition 2. *A* specialization *is an algebra homomorphism* $\varphi : \mathbb{K}[\mathbf{T}, \mathbf{X}] \to \widetilde{\mathbb{K}}[\mathbf{X}]$ *such that* $\varphi(\mathbb{K}[\mathbf{T}]) \subseteq \widetilde{\mathbb{K}}$ *and* $\varphi(X_i) = X_i$, $i = 1, \ldots, l$*. The* specialization associated with *a point* $t \in \widetilde{\mathbb{K}}^m$ *is defined by* $\varphi_t(p(\mathbf{T}, \mathbf{X})) := p(t, \mathbf{X})$*.*

Let $I = \langle f_1(\mathbf{T}, \mathbf{X}), \ldots, f_r(\mathbf{T}, \mathbf{X})\rangle \subseteq \mathbb{K}[\mathbf{T}, \mathbf{X}]$ be an ideal, let $<$ be a term ordering in $\mathbb{K}[\mathbf{T}, \mathbf{X}]$ and $G = \{g_1(\mathbf{T}, \mathbf{X}), \ldots, g_s(\mathbf{T}, \mathbf{X})\}$ be a Gröbner basis (GB) of I with respect to $<$.

The most practical question is to find $W \subseteq \widetilde{\mathbb{K}}^m$ such that $t \in W$ if and only if its associated specialization φ_t has the property that $\varphi_t(G)$ is a GB of $\varphi_t(I)$. Computationally speaking this is the most interesting aspect, because, once W is computed, every time we have a particular parameter vector we simply test if it belongs to W to verify if we have a GB or not.

There are at least three possible approaches: perform the initial Gröbner basis computation in $\mathbb{K}(\mathbf{T})[\mathbf{X}]$ (see [3], pp. 277, exercises 5 to 9), in $\mathbb{K}[\mathbf{T}][\mathbf{X}]$ (see [9]) or in $\mathbb{K}[\mathbf{T}, \mathbf{X}]$ ([7]). In the present work, the third one is chosen. Among the known results we cite Gianni's, Kalkbrener's and Montes' ones, which are briefly exposed below. We substantially adopt a mix of their global settings.

Gianni's approach: Given φ, we say a term ordering $>$ for $(\mathbf{T}, \mathbf{X})$ is φ-*admissible* if it is a block ordering with respect to $\mathbf{T}$ and $\mathbf{X}$, and $\mathbf{X} > \mathbf{T}$, i.e., if $>_X$ (resp. $>_T$) is the restriction of $>$ to the $\mathbf{X}$-variables (resp. to the $\mathbf{T}$-variables), then for all $A, C \in \mathbb{N}^l$ and for all $B, D \in \mathbb{N}^m$:

$$\mathbf{T}^B\mathbf{X}^A > \mathbf{T}^D\mathbf{X}^C \iff \mathbf{X}^A >_X \mathbf{X}^C \text{ or } (\mathbf{X}^A = \mathbf{X}^C \text{ and } \mathbf{T}^B >_T \mathbf{T}^D)$$

For $K \subset \mathbb{K}[\mathbf{T}, \mathbf{X}]$, $Lt(K) = \langle lt(f) \mid f \in K \rangle$, where lt is the leading term. Let

$$f(\mathbf{T}, \mathbf{X}) = \sum_{\beta=\beta_1,\beta_2} a_\beta \mathbf{T}^{\beta_1}\mathbf{X}^{\beta_2} = \sum_\alpha u_\alpha(\mathbf{T})\mathbf{X}^\alpha \in \mathbb{K}[\mathbf{T}, \mathbf{X}]$$

where in the second sum the power products of $\mathbf{X}$ are collected, when possible, and u_α are the corresponding polynomial "coefficients" in $\mathbf{T}$. Here $lt_1(f)$ will denote $u_A(\mathbf{T})\mathbf{X}^A$ where $\mathbf{X}^A$ is the biggest (according to $>_X$) $\mathbf{X}$-term in f, and $Lt_1(K) = \langle lt_1(f) \mid f \in K \rangle$.

Theorem 1. *([7]) Let $I \subset \mathbb{K}[\mathbf{T}, \mathbf{X}]$ be an ideal, φ a specialization and G a GB for I with respect to a φ-admissible ordering. If*

$$Lt(\varphi(I)) = \varphi(Lt_1(I)) \tag{1}$$

then $\varphi(G)$ is a GB for $\varphi(I)$.

Note that being G a GB of I, we can write $Lt_1(I) = \langle lt_1(g_1), \ldots, lt_1(g_s) \rangle$ and therefore we know the generators of the ideal on the right of (1), $\varphi(Lt_1(I))$:

$$\varphi(Lt_1(I)) = \langle \varphi(lt_1(g_1)), \ldots, \varphi(lt_1(g_s)) \rangle$$

However, we lack a finite representation of the ideal on the left $Lt(\varphi(I))$, unless we know *a priori* a GB of $\varphi(I)$, but finding it is indeed our purpose. In the zero-dimensional, univariate case $\varphi(G)$ is always a GB of $\varphi(I)$.

Kalkbrener's approach: he proposes a slightly different approach. The specialization map is $\varphi : R \to \mathbb{K}$, where R is a noetherian commutative ring ($\mathbb{K}[\mathbf{T}]$ in our case), and G a GB of I in $R[\mathbf{X}]$ with respect to any term ordering of $R[\mathbf{X}]$ (note that this approach requires the computation of a GB in $\mathbb{K}[\mathbf{T}][\mathbf{X}]$). The question is then: when is $\varphi(G)$ a GB of $\varphi(I)$? The answer is related to Gianni's approach. In [9] it is shown that (1) is equivalent to:

Assume that the g_i are ordered in such a way that there exists one $r \in \{0, \ldots, s\}$ with $\varphi(lt_\varphi(g_i)) \neq 0$ for $i \in \{1, \ldots, r\}$, and $\varphi(lt_\varphi(g_i)) = 0$ for $i \in \{r+1, \ldots, s\}$. Then, for every $i \in \{r+1, \ldots, s\}$ the polynomial $\varphi(g_i)$ is reducible to 0 modulo $\{\varphi(g_1), \ldots, \varphi(g_r)\}$.

Montes' approach: in [12] the author describes DISPGB algorithm (working in $R[\mathbf{T}][\mathbf{X}]$, R unique factorization domain) which computes all the reduced GB corresponding to the different possible cases for the parameters. It's a recursive procedure which branches when a "critical" (head) coefficient may be zero or not. A computation tree carrying on Buchberger calculations and current parameter conditions is constructed and analyzed. For every leaf the simplified corresponding condition (when it is not inconsistent) and the particular GB is presented. An important component is the procedure to successively simplify the conditions, both to detect as soon as possible an eventual inconsistency and to obtain a "normalized" presentation of them.

3 Using Hilbert Functions

Let $J \subseteq \mathbb{K}[Y_0, \ldots, Y_n] = \mathbb{K}[\mathbf{Y}]$ be an ideal. J is *homogeneous* if $f \in J$ implies that its homogeneous components are in J. Let $\mathbb{K}[\mathbf{Y}]_v$ denote the set of homogeneous polynomials of total degree v (including the zero polynomial). It's a $\mathbb{K}$-vector space of dimension $\theta(n, v) = \binom{n+v}{v}$. If J is homogeneous, $J_v = J \cap \mathbb{K}[\mathbf{Y}]_v$.

Definition 3. *The* Hilbert function *of J, $\mathcal{H}_J : \mathbb{N} \to \mathbb{N}$, is defined by $\mathcal{H}_J(v) = \dim_{\mathbb{K}}(\mathbb{K}[\mathbf{Y}]_v / J_v)$. The* Hilbert (Poincaré) series *of J is the formal power series*

$$H^J(y) = \sum_{v=0}^{\infty} \mathcal{H}_J(v)\, y^i$$

It is proved it can be expressed as a rational function, with denominator $D = (1-y)^n$. The numerator is the *Hilbert numerator* of J, HN_J (simply HN if J is clear from the context). If J is embedded in a polynomial ring with k more variables, $\mathcal{H}_J$ changing concerns only D. It becomes $(1-y)^{n+k}$. An efficient way to test Hilbert functions equality is simply to test equality on Hilbert numerators.

It is possible to define Hilbert function etc. for inhomogeneous ideals, too. Just replace $\mathbb{K}[\mathbf{Y}]_s$ with $\mathbb{K}[\mathbf{Y}]_{\leq s}$, the set of polynomials in $\mathbb{K}[\mathbf{Y}]$ of total degree at most s (including the zero polynomial).

3.1 The Homogeneous Case

Following [15] and [8], we can use the Hilbert function to check if the specialization of a GB of $J \subseteq \mathbb{K}[\mathbf{T}, \mathbf{X}]$ with respect to a block ordering for $\mathbf{X}$ and $\mathbf{T}$ variables is a GB of the specialized ideal. The main tools are the below reported theorem and its corollary, which are the starting point to tackle the problem.

Theorem 2. *([15]) Let $J \subseteq \mathbb{K}[\mathbf{Y}]$ be an ideal, $<$ a degree-compatible term ordering and $G = \{g_1, \ldots, g_s\} \subseteq J$. If $H_{Lt_<(J)} = H_{Lt_<(G)}$ then G is a GB of J with respect to $<$.*

Corollary 1. *([8]) Let J be a $(\mathbf{X})$-homogeneous ideal in $\mathbb{K}[\mathbf{T}, \mathbf{X}]$, G its GB with respect to a block $\mathbf{X}$-degree-compatible ordering $<$ in $\mathbb{K}[\mathbf{T}, \mathbf{X}]$ and φ the specialization map giving values to the parameters $\mathbf{T}$. Let $<_X$ be the restriction of $<$ to $\mathbb{K}[\mathbf{X}]$, $H_{Lt_{<_X}(G)}$ the Hilbert function of the ideal generated by the $\mathbf{X}$-leading power products of G and $H_{Lt_{<_X}(\varphi(G))}$ the Hilbert function of the ideal generated by the leading power products of $\varphi(G)$. If $H_{Lt_{<_X}(G)} = H_{Lt_{<_X}(\varphi(G))}$ then $\varphi(G)$ is a GB of $\varphi(J)$ with respect to $<_X$.*

In [8] the key point was to compute a 'generic' GB G of $J \subseteq \mathbb{K}[\mathbf{T}, \mathbf{X}]$ with respect to a $\mathbf{X}$-degree-compatible term ordering and use G to derive the "generic" Hilbert function $\mathcal{H}_1$ of J considering only $\mathbf{X}$ variables. Substituting specific values for the parameters $\mathbf{T}$, the specialization of G is a GB if its Hilbert function $\mathcal{H}_2$ agrees with $\mathcal{H}_1$. Viewing it graphically:

$$\begin{tikzcd} F \arrow[r] & G \arrow[r] \arrow[d, Rightarrow] & \varphi(G) \arrow[d, Rightarrow] \\ & \mathcal{H}_1 & \mathcal{H}_2 \end{tikzcd}$$

3.2 The Inhomogeneous Case

The above results are no longer true when applied to non $\mathbf{X}-$homogeneous ideals J, but they may be extended. Instead of changing the definition of Hilbert function as indicated above, we keep it as it is, but consider enlarged working rings $\mathbb{K}[\mathbf{T}, \mathbf{X}, H]$ and $\mathbb{K}[\mathbf{X}, H]$, where H is a new variable to homogenize polynomials.

Definition 4. *Let* $\mathbf{Y} = \{Y_1, \ldots, Y_n\}$*:*

1. *For a polynomial* $f \in \mathbb{K}[\mathbf{Y}]$ *with* $\deg(f) = d$*, its* homogenization *is the homogeneous polynomial* $f^h \in \mathbb{K}[\mathbf{Y}, H]$ *such that*

$$f^h(\mathbf{Y}, H) = H^d f(Y_1/H, \ldots, Y_n/H)$$

2. *For a homogeneous polynomial* $f \in \mathbb{K}[\mathbf{Y}]$ *and a variable* $Y_i \in \mathbf{Y}$*, its* dehomogenization *gives* ${}^h f \in \mathbb{K}[\mathbf{Y}'] = \mathbb{K}[Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_n]$ *such that*

$${}^h f(\mathbf{Y}') = f(Y_1, \ldots, Y_{i-1}, 1, Y_{i+1}, \ldots, Y_n)$$

It is easy to see that $f = {}^h(f^h)$, if the dehomogenization is done with respect to the homogenizing variable. Our idea is what we call the *Hilbert test*:

Step I: Compute a GB G of J with respect to a block $\mathbf{X}$-degree-compatible ordering $<$ in $\mathbb{K}[\mathbf{T}, \mathbf{X}]$.

Step II: Homogenize with respect to $\mathbf{X}$ the polynomials in G using H variable (smaller than $\mathbf{X}$ variables, bigger than $\mathbf{T}$ ones), letting G^h be this new set of polynomials. Let $\mathcal{H}_1$ be the Hilbert function of $Lt_{<'_X}(G)$ with $<'_X$ as in corollary 1, but opportunely extended to consider also H.

Step III: Specialize the parameters $\mathbf{T}$ using φ on G^h, obtaining $\varphi(G^h)$. Let $\mathcal{H}_2$ be the Hilbert function of $Lt_{<'_X}(\varphi(G^h))$. If $\mathcal{H}_2 = \mathcal{H}_1$ then φ is good.

Proposition 1. *The Hilbert test is a sufficient condition for* φ *to be good.*

Proof. G^h is a homogeneous system, therefore if $\mathcal{H}_1 = \mathcal{H}_2$ then $\varphi(G^h)$ is a GB thanks to the homogeneous case results. Dehomogenization preserves the Gröbner properties, therefore ${}^h\varphi(G^h) = \varphi(G)$ is Gröbner. □

A specialized basis may be Gröbner but not necessarily reduced. Note that, because of the block $\mathbf{X}$-degree-compatible term ordering condition, G and G^h have the same $\mathbf{X}$-heads, and is therefore possible to compute HN_J indifferently before or after homogenizing, because adding H variable gives no problem. Viewing everything graphically,

$$\boxed{\begin{tikzcd}
\mathcal{H}_1 & G^h \arrow[l, Rightarrow] \arrow[r] & \varphi(G^h) \arrow[r, Rightarrow] \arrow[d] & \mathcal{H}_2 \\
F \arrow[r] & G \arrow[u] & {}^h\varphi(G^h) = \varphi(G) &
\end{tikzcd}}$$

To simplify notation, we define a new set of variables $\mathbf{Z}$ as follows:

$$\mathbf{Z} = \begin{cases} \mathbf{X} & \text{if } F \text{ is } \mathbf{X}\text{-homogeneous} \\ (\mathbf{X}, H) & \text{otherwise} \end{cases}$$

Multindexes exponents $\alpha_{i,j}$ length changes accordingly, and the number of variables eventually increases by one, but will still be indicated with l.

3.3 Computing a Good Specialization Conditions Set

Let J be an ideal in $\mathbb{K}[\mathbf{T},\mathbf{Z}]$, $G=\{g_1,\ldots,g_s\}$ its GB with respect to a block $\mathbf{Z}$–degree–compatible ordering $<$ in $\mathbb{K}[\mathbf{T},\mathbf{Z}]$. Every polynomial g_i has the following structure

$$g_i(\mathbf{T},\mathbf{Z})=\sum_{j=0}^{m_i-1} u_{i,j}(\mathbf{T})\,\mathbf{Z}^{\alpha_{i,j}}, \qquad \alpha_{i,0}>_Z \alpha_{i,1}>_Z \cdots$$

with $<_Z$ restriction of $<$ to $\mathbb{K}[\mathbf{Z}]$. For notational convenience we define $u_{i,m_i}(\mathbf{T})=1$, $\mathbf{Z}^{\alpha_{i,m_i}}=\mathbf{0}$ for each $i=1,\ldots,s$. Let $L=[n_1,\ldots,n_s]$ be a list of s integers with $0\le n_i\le m_i$, $L[i]=n_i$, $\mathcal{L}$ the set of all such lists and $L_0=[0,\ldots,0]$. We introduce a partial ordering $\ge$ in $\mathcal{L}$:

$$[n_1,\ldots,n_s]\ge[k_1,\ldots,k_s] \quad\Longleftrightarrow\quad n_1\ge k_1,\ldots,n_s\ge k_s$$

The specialization condition defined by L is given by the $\mathbf{T}$ satisfying

$$\{u_{i,0}(\mathbf{T})=0,\ldots,u_{i,n_i-1}(\mathbf{T})=0\ ,\ u_{i,n_i}(\mathbf{T})\neq 0,\ 1\le i\le s\}$$

We call $C(L)=C(u_{i,j})$ this set. The notation permits to express conditions as lists of common length s, such that there is at least one "$\neq 0$" component.

Using proposition 1, for any specialization φ verifying $C(L)$, if the Hilbert functions $\mathcal{H}_L$ of $\mathcal{Z}_L=\langle \mathbf{Z}^{\alpha_{i,n_i}}:1\le i\le s\rangle$ and $\mathcal{H}_0$ of $\mathcal{Z}_0=\langle \mathbf{Z}^{\alpha_{i,0}}:1\le i\le s\rangle$ coincide, then $\varphi(G)$ is a GB of $\varphi(J)$ (with respect to $<_Z$).

Definition 5. *A condition list $L=[n_1,\ldots,n_s]$ is* good *if $\mathcal{H}_L=\mathcal{H}_0$,* bad *otherwise. It is* empty *if $C(L)=\emptyset$.*

This suggests a nice way to build a set of conditions giving good specializations for the GB of J, already computed in $\mathbb{K}[\mathbf{T},\mathbf{X}]$: namely, non empty ones given by good lists.

There are in general $M=\prod_{i=1}^{s}(m_i+1)$ cases, but it is often possible to avoid some of them. It is in fact evident that if there are some $\widetilde{m}_{ij}\le m_i-1$ such that $u_{i,\widetilde{m}_{ij}}(\mathbf{T})$ is (a not zero) constant, the i^{th} entry in any condition list will be necessarily $\leqslant \widetilde{m}_i=\min\{\widetilde{m}_{ij}\}$. In this way it is possible to reduce the total number of cases from M to $\widetilde{M}$, substituting the factors in the expression for M with $(\widetilde{m}_i+1)$. If $\widetilde{m}_i\ll m_i$ (for example 0) the advantage is relevant.

In any case, L_0 surely provides a good specialization condition. Next, the vanishing of some $\mathbf{T}$–coefficients into only one g_j polynomial is analyzed, then in two polynomials at the same time, and so on.

Let $\overline{m}_i=\min\{\widetilde{m}_i,m_i\}$. We can schematically represent the part of the generic GB which is interesting for our analysis as follows, with $u_{i,\overline{m}_i}(\mathbf{T})$ not zero constant for each i.

$$\begin{array}{r|lllll|}
\hline
g_1(\mathbf{T},\mathbf{Z}): & u_{1,0}(\mathbf{T}) & u_{1,1}(\mathbf{T}) & u_{1,2}(\mathbf{T}) & \ldots & u_{1,\overline{m}_1}(\mathbf{T}) \\
g_2(\mathbf{T},\mathbf{Z}): & u_{2,0}(\mathbf{T}) & u_{2,1}(\mathbf{T}) & u_{2,2}(\mathbf{T}) & \ldots & u_{2,\overline{m}_2}(\mathbf{T}) \\
g_3(\mathbf{T},\mathbf{Z}): & u_{3,0}(\mathbf{T}) & u_{3,1}(\mathbf{T}) & u_{3,2}(\mathbf{T}) & \ldots & u_{3,\overline{m}_3}(\mathbf{T}) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
g_s(\mathbf{T},\mathbf{Z}): & u_{s,0}(\mathbf{T}) & u_{s,1}(\mathbf{T}) & u_{s,2}(\mathbf{T}) & \ldots & u_{s,\overline{m}_s}(\mathbf{T}) \\
\hline
\end{array}$$

We must analyze all compatible possibilities selecting **X**-terms among the boxed "coefficients" in a combinatoric way - one per line - performing Hilbert functions comparisons and keeping lists giving good conditions. Later we may discard the empty (unsatisfiable) ones and simplify the presentation of non empty ones.

For example, with $F = \{x^4, y^4, bx^3y+cx^2y^2+dxy^3, ax^2y^2\}$, where $\mathbf{X} = \{x, y\}$ and $\mathbf{T} = \{a, b, c, d\}$, using a block DRL ordering the resulting good lists in $\mathcal{L}$ are

```
[0,0,0,0,0,0,0]  (1)     [0,0,0,0,1,1,0]  E      [0,0,2,0,0,2,0]  E
[0,0,0,0,0,0,1]  (2)     [0,0,0,0,1,1,1]  E      [0,0,2,0,0,2,1]  E
[0,0,0,0,0,1,0]  (3)     [0,0,0,0,1,2,0]  E      [0,0,2,0,1,0,0]  E
[0,0,0,0,0,1,1]  E       [0,0,0,0,1,2,1]  (5)    [0,0,2,0,1,0,1]  E
[0,0,0,0,0,2,0]  E       [0,0,2,0,0,0,0]  E      [0,0,2,0,1,1,0]  (6)
[0,0,0,0,0,2,1]  E       [0,0,2,0,0,0,1]  E      [0,0,2,0,1,1,1]  E
[0,0,0,0,1,0,0]  (4)     [0,0,2,0,0,1,0]  E      [0,0,2,0,1,2,0]  E
[0,0,0,0,1,0,1]  E       [0,0,2,0,0,1,1]  E      [0,0,2,0,1,2,1]  E
```

where E=empty and numbers in brackets correspond to the following conditions

(1)		$a, b, c^2 - bd, c, d \neq 0$
(2)	$d = 0;$	$a, b, c \neq 0$
(3)	$c = 0;$	$a, b, d \neq 0$
(4)	$c^2 - bd = 0;$	$a, b, c, d \neq 0$
(5)	$c, d = 0;$	$a, b \neq 0$
(6)	$c, b = 0;$	$a, d \neq 0$

The possible combinations are not at all trivial, even in very easy cases. Generalizing the approach, to compute good specialization conditions one may choose among two "dual" possibilities, considering the order in which exclusion criteria are analyzed.

$(\mathbf{Z}, \mathbf{T})$ Look for good lists and among them for non empty ones (above example). In this case we first use the Hilbert test to discard bad lists, and then apply simplification techniques to discard the empty ones. Here the work has to be done first in the domain of **Z** variables, and eventually in $\mathbb{K}[\mathbf{T}]$.

$(\mathbf{T}, \mathbf{Z})$ Vice versa, look for compatible specialization conditions, and for each non empty one, test if it's good.

The two approaches lead to exactly the same result, but which is the best one ? In a completely general situation, $(\mathbf{Z}, \mathbf{T})$ requires the computation of say M Hilbert tests, and L_g calls to our procedure for conditions simplification, one for each good list. $(\mathbf{T}, \mathbf{Z})$ requires instead the computation of all C_L possible $C(L)$, and of Hilbert tests for all the E_g non empty ones. C_L is easily obtained considering that each coefficient may be equal to or different from zero, and therefore there are 2^{m_i} possibilities for each polynomial in the basis.

It is very difficult to give general estimates (apart trivial ones) for L_g and E_g. For what concerns complexity, indicating the maximum computational cost of a condition simplification with C_S and of a Hilbert test with C_H, we have

$$C(\mathbf{Z},\mathbf{T}) = C_H \cdot \left(\prod_{i=1}^{s}(m_i+1)\right) + C_S \cdot L_g \quad ; \quad C(\mathbf{T},\mathbf{Z}) = C_S \cdot 2^{\sum_i^s m_i} + C_H \cdot E_g$$

Or, more roughly, if $\hat{m} = \max\{m_i\}$,

$$C(\mathbf{Z},\mathbf{T}) = C_H \cdot (\hat{m}+1)^s + C_S \cdot L_g \qquad ; \quad C(\mathbf{T},\mathbf{Z}) = C_S \cdot 2^{s\hat{m}} + C_H \cdot E_g$$

However, in real-life systems the situation is never so general: the number of parameters is much smaller than M, and some $\mathbf{Z}-$term is divisible by other ones. These facts often contribute to reduce the number of cases to be treated. We propose two criteria to prune the tree of possible configurations to analyze.

4 Pruning Criteria

We indicate in this section two criteria to limit the number of cases to be treated. The former refers to $\mathbf{Z}$ variables, the latter to $\mathbf{T}$. Both remove many lists using informations concerning a single one, and can be defined as *global*.

Unfortunately, the very interesting sentence "if L is bad and $L' > L$ then L' is bad, too" is generally false. In fact, consider the ideal $I = \langle F \rangle$ for the above introduced F. Its GB is $G = F \cup \{(c^2 - bd)x^2y^3, cx^3y^2 + dx^2y^3, dx^3y^3\}$, with "generic" $\mathbf{Z}$-monomial ideal $\langle x^4, y^4, x^2y^2, x^3y\rangle$. The generic Hilbert numerator for the condition list $L = [0,0,0,0,0,0,0]$ is $HN(x) = x^6 + 2x^5 - 4x^4 + 1$.

Consider a specialization φ_1 such that $\varphi_1(a) \neq 0$, $\varphi_1(b) = 0$, $\varphi_1(c) \neq 0$, $\varphi_1(d) \neq 0$ (condition list $L' = [0,0,1,0,0,0,0]$). Then $\varphi_1(Lt_{<_Z}(I)) = \langle x^4, y^4, x^2y^2\rangle$, and we have $HN'(x) = 2x^6 - 3x^4 + 1$. Here $HN \neq HN'$, therefore L' is bad.

Now take φ_2 such that $\varphi_2(a) \neq 0$, $\varphi_2(b) = \varphi_2(c) = 0$, $\varphi_2(d) \neq 0$ for $L'' = [0,0,2,0,1,1,0]$. Then $\varphi_2(Lt_{<_Z}(I)) = \langle x^4, y^4, x^2y^2, xy^3\rangle$, and for symmetry with respect to the "generic" situation it is clear that $HN'' = HN$, then L'' is good.

We then have $L < L' < L''$, with L and L'' good and L' bad. Note that, when there are repetitions, substituting a term with a smaller one does not necessarily lower the Hilbert function: in general everything can happen. Moral: the above sentence indicates a *local* condition.

$\boxed{\textbf{First criterion}}$ A very simple consideration: let L be an indexes list and

- $d_L = \min\{\deg(\mathbf{Z}^{\alpha_{i,L[i]}}) \mid i = 1,...,s\}$ the smallest degree of the $\mathbf{Z}$-terms indicated by L.
- G_L the set of polynomials of G^h with $\mathbf{Z}$-degree d_L, and s_L its cardinality.
- $G^L_{d_L}$ the GB of $\mathcal{Z}_L \cap \mathbb{K}[\mathbf{Z}]_{d_L}$, and h_L ($\leq s_L$) its cardinality.

where d_0, $\mathcal{H}_0$, G_0, s_0, $G^0_{d_0}$, h_0 correspond to d_{L_0}, $\mathcal{H}_{L_0}$, G_{L_0}, s_{L_0}, $G^{L_0}_{d_{L_0}}$, h_{L_0}.

If $0 < v < d_0$, then $\mathcal{H}_L(v) = \mathcal{H}_0(v) = \binom{l+v-1}{v}$ for each possible list L (no monomial enters the game yet). The case $v = d_0$ is the first one for which the Hilbert function depends on the monomial ideal, and we have

$$\mathcal{H}_0(d_0) = \binom{l+d_0-1}{d_0} - h_0$$ (the only monomials to be discarded are exactly the ones in $G^0_{d_0}$)

For $v > d_0$ the computation is more involved, and we won't detail more. It is however clear that if we want to have $\mathcal{H}_L = \mathcal{H}_0$ it is necessary that (at least) $\mathcal{H}_L(d_0) = \mathcal{H}_0(d_0)$, and the *only* way to make it happen is that $G^L_{d_0}$ (the GB of $\mathcal{Z}_L \cap \mathbb{K}[\mathbf{Z}]_{d_0}$) has $h_L = h_0$ elements. Due to possible repetitions it may happen that $G^L_{d_0}$ has less or more elements. If it is so, we already know the Hilbert test will fail, and it is useless to continue analyzing the L case.

Summing all up, ${}^{\#}G^L_{d_0} \neq {}^{\#}G^0_{d_0} \Rightarrow L$ is bad.

Suppose that L is bad *because of this criterion*. Then we can discard all lists in $\mathcal{L}$ with indexes corresponding to polynomials in F_0 equal to those in L.

$\boxed{\textbf{Second criterion}}$ It uses the $u_{i,j}(\mathbf{T})$. Each condition except $C(L_0)$ is like:

$$\{p_1(\mathbf{T}) = 0, \ldots, p_z(\mathbf{T}) = 0, q_1(\mathbf{T}) \neq 0, \ldots, q_{n_z}(\mathbf{T}) \neq 0\} \tag{2}$$

where $z, n_z \geq 1$ and $p_i, q_i \in \mathbb{K}[\mathbf{T}]$ are non constant. We refer to the polynomials

- p_i as components of the *zero part* $Z = Z(L) = Z(p_1, \ldots, p_z)$,
- q_j for the *non zero part* $NZ = NZ(L) = NZ(q_1, \ldots, q_{n_z})$.

Definition 6. *A list $L \in \mathcal{L}$ is* inappropriate *if $C(Z(L)) = \emptyset$.*

It is clear that (good or bad) inappropriate lists are of no interest at all, and we can avoid to consider them. We are now ready to state the following

Lemma 1. *If L is an inappropriate condition list for G then every condition list L' such that $L' > L$ is also inappropriate.*

Proof. Let $Z = Z(L) = Z(p_1, \ldots, p_k)$ be the zero part of L. For $L' > L$ we have $Z' = Z(L') = Z(p_1, \ldots, p_k, p_{k+1}, \ldots, p_{k'}) = Z \cap Z(p_{k+1}, \ldots, p_{k'}) = \emptyset \cap Z(p_{k+1}, \ldots, p_{k'}) = \emptyset$. □

As soon as an inappropriate list is found, we can then directly discard it *and all the greater ones*, avoiding to consider many cases. The drawback is that one may have to compute incrementally various intermediate Gröbner bases.

5 Conditions Simplification

As we have described in the previous section, we have a set of good specialization conditions in the non trivial case ($\varphi(I) \neq \langle 1 \rangle$). In the following paragraphs we'll show how to simplify a condition or recognize that it is empty. Obviously $(Z(1), NZ(q_1, ..., q_m)) = \emptyset$.

How much can a simplified condition be complicated? In other words, which are the limitations for $z = \#Z$ and $n_z = \#NZ$ in a simplified condition ?

[**NZ**] at most one coefficient for each polynomial may be imposed to be different from zero (when it is not constant). Therefore the number of polynomials in G is an upper bound for n_z. We obtain $0 \leq n_z \leq s$.

[**Z**] Let $\delta = \max\{\deg(p_i) \mid p_i \in Z\}$: suppose the p_i are a GB for Z. The worst possible case is that for all $\mathbf{T}^\beta$ with $|\beta| = \delta$ there is p_j with $lt(p_i) = \mathbf{T}^\beta$ – this means all p_i have degree δ. The limitation is then $0 \leq z \leq \binom{m+\delta-1}{\delta} = \overline{z}$.

5.1 One Parameter Case

It stands out because it is possible to eliminate NZ. Let $\mathbf{T} = t$: the following propositions justify the basic steps of an algorithm for conditions treatment.

Proposition 2.

1. $(p, q) = 1 \Rightarrow (Z(p), NZ(q)) \equiv Z(p)$

2. $(p, q) = d, \deg(d) > 0 \Rightarrow (Z(p), NZ(q)) \equiv (Z(p/d), NZ(q)) \equiv (Z(p/d), NZ(d, q/d))$

3. $p = (p_1, \ldots, p_k) \Rightarrow (Z(p_1, \ldots, p_k), NZ(q_1, \ldots, q_m)) \equiv (Z(p), NZ(q_1, \ldots, q_m))$

4. $(p, q_1) = 1 \Rightarrow (Z(p), NZ(q_1, q_2, ..., q_m)) \equiv (Z(p), NZ(q_2, ..., q_m))$.
$(p, q_1) = d \neq 1 \Rightarrow (Z(p), NZ(q_1, ..., q_m)) \equiv (Z(p/d), NZ(d, q_1/d, ..., q_m))$.

Proof. *1. Suppose p and q are co-prime. Then, by Bezout's identity, there are* $a(t)$, $b(t) \in \mathbb{K}[t]$ *such that* $1 = a(t)p(t) + b(t)q(t)$. *Let* $p(r) = 0$. *Setting* $t = r$

$$1 = a(r)p(r) + b(r)q(r) = a(r) \cdot 0 + b(r)q(r) = b(r)q(r)$$

That is, q is automatically not zero on $Z(p)$: $NZ(q)$ *can be removed.*

2. We have $p(t) = p_1(t)d(t)$ *and* $q(t) = q_1(t)d(t)$, *where* p_1 *and* q_1 *are the cofactors. Now* $p(t) = 0 \Leftrightarrow (d(t) = 0$ *or* $p_1(t) = 0)$, *and* $q(t) \neq 0 \Leftrightarrow (d(t) \neq 0$ *and* $q_1(t) \neq 0)$. *Then it is not possible that* $d(t) = 0$, *and we can substitute p with* p_1 *(and eventually q with its found partial factorization).*

3. Obvious, from the fact that r is a root of $p_1, \ldots, p_k$ *if and only if it is a root of p. If* $p(t) = 1$ *the condition is clearly empty.*

4. Simply apply cases 1 and 2. □

Applying case 3 we obtain a singleton zero part, and (if it is not $Z(1)$) we may repeatedly apply 4 (possibly 1) either to cancel a polynomial in NZ or to lower degrees. Both conditions make the algorithm terminate, and at the end, if the condition was not discarded, it simply becomes $Z(p_{\text{final}})$.

5.2 Several Parameters Case

When dealing with many parameters, the above approach cannot be straightforwardly followed. If there are by chance polynomials which contain one variable, we can follow the above explained approach, but in general proposition 1 and consequently the first of proposition 4 are no more true. A possible strategy may be a cycle with the following steps, ending when there are no more modifications.

1. Compute a GB of the zero part Z (which we call again Z). If it is trivial, the condition is discarded, otherwise...
2. Compute the normal forms (with respect to Z) $\overline{q}_j$ of polynomials in NZ.
3. Perform the following tests and actions: if one of the $\overline{q}_j$ is a non zero constant it is removed from NZ, because it's redundant. If it is 0, the whole condition is discarded for incompatibility, otherwise...
4. Continue using proposition 2 for all pairs $p \in Z$, $q \in NZ$ for which it applies. If there is at least one such pair, repeat step 3 taking into account the new introduced polynomials d, p/d and q/d, otherwise terminate.

If the coefficients represent (hyper-)surfaces in generic position, every time we add one of them to Z the dimension of the associated variety lowers by at least one. In this fortunate case $z \leq m$, but the general bound remains $\overline{z}$.

6 Analysis Refinement and Hilbert Stratification

Thanks to the previous sections results, we can partition the parameter space into two sets: the *good* one - values satisfying the conditions corresponding to good lists - and its complement. Note that, Hilbert test giving a *sufficient* condition, may be that some points out of the good set have associated specializations mapping G into a GB, but this still has to be verified.

In order to refine the obtained results, we have to continue and see what really happens for every bad list. More precisely, the idea is to proceed as follows.

- Let L be a bad list for G^h with $C(L) \neq \emptyset$, and $\mathcal{H}_1$ as specified in the Hilbert test. Specialize (partially, in general) G^h setting to zero all the $u_{i,j}(\mathbf{T}) \in Z(L)$ and reducing the others modulo the GB of $Z(L)$. We can see this as the application of a certain specialization φ, which in general we cannot associate to a well determined point of $C(L)$.
- Let $\varphi(G^h)$ be the result, $\mathcal{H}_2$ the Hilbert function computed using its head $\mathbf{Z}$-terms, ($\mathcal{H}_1 \neq \mathcal{H}_2$ by hypothesis). Up to now, we don't know if $\varphi(G^h)$ is a GB or not, and we apply Buchberger algorithm again, obtaining G'.
- Let $\mathcal{H}_3 = H_{Lt_{<_Z}\langle G' \rangle}$. Then
  - $\star$ $\mathcal{H}_3 = \mathcal{H}_1 \Rightarrow \varphi(G^h)$ is not Gröbner, discard L (but φ preserves Hilbert function).
  - $\star$ $\mathcal{H}_3 = \mathcal{H}_2 \Rightarrow \varphi(G^h)$ is Gröbner. Then $C(L)$ is a conditions set good for specialization. The obtained basis will have a different Hilbert function with respect to the "generic" case, but it remains Gröbner. We could refer to this as a lucky case in an unlucky environment.
  - $\star$ $\mathcal{H}_3 \neq \mathcal{H}_1$, $\mathcal{H}_3 \neq \mathcal{H}_2 \Rightarrow \varphi(G^h)$ is not Gröbner, discard L.

At this point one could be interested to see what happens to Hilbert function. It is possible to continue, performing recursively the complete analysis of section 3.3. Replace G^h with G' and $\mathcal{H}_1$ with $\mathcal{H}_3$, taking also care of the conditions $C(L)$. Identify new good and bad lists, etc.

Graphically, we can represent everything as:

$$\begin{tikzcd} G^h \arrow[r] \arrow[d, Rightarrow] & \varphi(G^h) \arrow[r] \arrow[d, Rightarrow] & G' \arrow[ll, bend right] \arrow[d, Rightarrow] \\ \mathcal{H}_1 & \mathcal{H}_2 & \mathcal{H}_3 \end{tikzcd}$$

6.1 A More General View

In a more general setting, the above approach, slightly modified, can be used to compute effectively the so-called *Hilbert stratification* of the variety as seen in $\mathbf{Z}$ variables and parametrized by $\mathbf{T}$. Having $\mathcal{L}$, it becomes almost natural to use a divide-and-conquer approach, in which every list is empty or corresponds to a certain value of the Hilbert function. It is then possible to partition the whole parameter space into subsets S_i such that the Hilbert function is constant over each one. More precisely, the complete output that can be obtained is a sequence of pairs $O = \{(H_i, C_i) \ : \ i \in \mathcal{I}\}$ such that a specialization associated to a point satisfying conditions C_i produces a specialized ideal with Hilbert function $\mathcal{H}_i$.

Figures 1 and 2 show the main functions. The code can clearly be optimized, but we don't expose here all the technical details. The functions isGood(L, G) and isGood(G_1, G_2) check for equality for $\mathcal{H}_L$, $\mathcal{H}(G)$, $\mathcal{H}(G_1)$, $\mathcal{H}(G_2)$, respectively.

```
Input  : $F = \{f_1, ..., f_r\} \in \mathbb{K}[\mathbf{T}, \mathbf{X}]^r$
Output : A pairs list $O = \{(H_i, C_i)\}$
$G$ = computeGBOf($F$);
if ( isNotHomogeneous($F$) )
   $G$ = homogenizeWithRespToX($F$);
$O$ = completeAnalysis($G$, $(0, 1)$);
output $O$;
```

Fig. 1. Stratification algorithm

```
Input  : $F = \{f_1, ..., f_r\} \in \mathbb{K}[\mathbf{T}, \mathbf{X}]^r$
       : A condition $C$
Output : A list of pairs $O = \{(H_i, C_i)\}$

$O = \emptyset$; $\mathcal{L}$ = setOfAllPossibleLists($F$);
while ( isNotEmpty($\mathcal{L}$) ) do
 $L$ = extractAnElement($\mathcal{L}$);
 $C' = C(L) \wedge C$;
 if ( isInappropriate($C'$) )
   $\mathcal{L}$ = removeAllListsGreaterThan($L$, $\mathcal{L}$);
 else
 if ( isGood($L$, $F$) )
   $O = O \cup (\mathcal{H}_L, C')$;
 else
   $F_{\mathrm{spec}}$ = specializeAccordingTo($F$, $C'$);
   $F'$ = computeGBOf($F_{\mathrm{spec}}$);
   if (isGood($F$, $F'$) or isGood($F_{\mathrm{spec}}$, $F'$))
     $O = O \cup (H(F'), C')$;
   else
     $O = O \cup$ completeAnalysis($F', C'$);
output $O$;
```

Fig. 2. completeAnalysis function

removeAllListsGreaterThan$(L,\mathcal{L})$ realizes lemma 1; specializeAccordingTo(F, C) maps F into $\varphi(F)$, where φ has the behaviour explained above. If C_1, C_2 are two conditions, their conjunction is indicated with $C_1 \wedge C_2 = (Z(C_1) \cap Z(C_2), NZ(C_1) \cap NZ(C_2))$.

This is very close to the approach in [17,18] where Comprehensive Gröbner bases are defined and computed, and in [12], where you can find an algorithm to obtain the conditions and the corresponding specialized bases (see 2).

7 Conclusions

We have shown how to analyze parametric polynomial systems using as main tool the properties of GB under specialization.We presented a technique based on Hilbert function use and simplification of the result in an efficient way. The Hilbert stratification of a parametric variety may be computed avoiding many GB computations, and this permits to obtain efficiently many informations on the structure and topology of the specialized varieties.

References

1. W. W. Adams, P. Loustaunau. *An Introduction to Gröbner Bases.* Graduate Studies in Mathematics, Volume 3, AMS, 1994.
2. T. Becker, V. Weispfenning *Gröbner Bases: A Computational Approach to Commutative Algebra.* Graduate Studies in Mathematics, Volume 141, Springer Verlag, 1993 (second edition, 1998).
3. D. Cox, J. Little, D. O'Shea. *Ideals, Varieties, and Algorithms.* Springer–Verlag, 1991 (second corrected edition, 1998).
4. D. Duval. *Evaluation dynamique et clôture algébrique.* Journal of Pure and Applied Algebra 99, 267–295, 1995.
5. FRISCO. *A Framework for Integrated Symbolic/Numeric Computation.* ESPRIT Project LTR 21024, 1996–1999, European Union.
6. A. Galligo, L. Poittier, C. Traverso. *Greatest Easy Common Divisor and standard basis completion algorithms.* ISSAC'98. Lecture Notes in Computer Science 358, 162–176, 1988.
7. P. Gianni. *Properties of Gröbner basis under specializations.* European Conference on Computer Algebra, Leipzig, GDR, 1987 (J. H. Davenport, ed.). Lecture Notes in Computer Science 378, 293–297, 1987.
8. M.J. Gonzalez Lopez, L. Gonzalez Vega, C. Traverso, A. Zanoni *Gröbner Bases Specialization through Hilbert Functions: The Homogeneous Case.* SIGSAM Bulletin, Communications in Computer Algebra. Volume 34 (1) Issue 131, 1–8, 2000.
9. M. Kalkbrener. *On the stability of Gröbner bases under specializations.* Journal of Symbolic Computation 24 (1), 51–58, 1997.
10. E. Kruppa. *Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung* Sitz.-Ber. Akad. Wiss., Wien, math. naturw. Abt. IIa, 122:1939-1948, 1913.
11. S.J. Maybank, O.D. Faugeras. *A Theory Of Self-Calibration Of A Moving Camera* IJCV (8),123–151, 1992
12. A. Montes. *A New Algorithm for Discussing Gröbner Bases with Parameters* Journal of Symbolic Computation 33 (2), 183-208, 2002.
13. M. Moreno–Maza. *Calculs de Pgcd au–dessus des Tours d'Extensions Simples et Résolution des Systèmes d'Équations Algébriques.* Doctoral Thesis, Université Paris 6, 1997.
14. W. Y. Sit. *An algorithm for solving parametric linear systems.* Journal of Symbolic Computation 13, 353–394, 1992.
15. C. Traverso. *Hilbert Functions and the Buchberger Algorithm.* Journal of Symbolic Computation 22 (4), 355–376, 1996.
16. B. L. van der Waerden. *Modern Algebra II.* Third edition. F. Ungar Publishing Co., NY, 1950.

17. V. Weispfenning. *Comprehensive Gröbner Bases.* Journal of Symbolic Computation, 14 (1), 1–30, 1992.
18. V. Weispfenning. *Canonical Comprehensive Gröbner bases.* Proceedings ISSAC 2002. ACM-Press, 270–276, 2002.
19. A. Zanoni *Numerical Gröbner Bases.* Tesi di dottorato, Università di Firenze, 2003.

Appendix: Examples

We present here some example for good conditions searching. Two used in [17], other two regarding ellipsoids intersection and finally a "skew" application on the *kruppa* system [10]. The term ordering is always a block DRL on unknowns and parameters.

A.1 Weispfenning's Tests

$F = \{vxy + ux^2 + x, uy^2 + x^2\}$, u, v parameters: we obtain the conditions

$$C_1 = \{v = 0\ ,\ u \neq 0\} \quad , \quad C_2 = \{u, u^3 + v^2, v \neq 0\}$$

$F = \{X_4 - (a_4 - a_2), X_1 + X_2 + X_3 + X_4 + (a_1 + a_3 + a_4), X_1X_3 + X_1X_4 + X_2X_3 + X_3X_4 - (a_1a_4 + a_1a_3 + a_3a_4), X_1X_3X_4 - a_1a_3a_4\}$, a_1, a_2, a_3, a_4 parameters:

$$C_1 = \{a_2 - a_4 = 0\ ,\ a_1a_3a_4 \neq 0\} \quad , \quad C_2 = \{a_2 - a_4 \neq 0\}$$

A.2 Ellipsoids

The first example represents the intersection of two fixed ellipsoids E_1 and E_2 centered in the origin having axes parallel to the principal ones, with a third one E_3. It has the same properties, but its center and eccentricities are parametric. For each ellipsoid, the canonical form of the equation $\frac{(x-a)^2}{A^2} + \frac{(y-b)^2}{B^2} + \frac{(z-c)^2}{C^2} = 1$ is transformed into $f_i(x, y, z)/A^2B^2C^2 = 0$, and just the numerator is considered.

Obviously, in this one and in the following example $A, B, C \neq 0$, and this fact – not codified in the equations – can be used to simplify the conditions.

$$F = \left\{ \begin{array}{ll} f_1 = x^2 + 4y^2 + 4z^2 - 4 & f_3 = (x-a)^2B^2C^2 + (y-b)^2A^2C^2 + \\ f_2 = 4x^2 + y^2 + z^2 - 4 & \quad (z-c)^2A^2B^2 - A^2B^2C^2 \end{array} \right\}$$

We obtain three final conditions:

$A(B^2 - C^2) \neq 0$: as we may have expected looking at f_1, f_2, "problems" (from a 0-dimensional to 1-dimensional system) arise when even E_3 has equal coefficients ($B = \pm C$) for y^2 and z^2.

$b, a, B^2 - C^2 = 0\ ;\ cAC \neq 0$: similarly as above. Note that the center third coordinate value is critical.

$aB, cA, bA, aA, A(B^2 - C^2) = 0$; $C(5A^2C^2 - 4A^2 - 4B^2) \neq 0$ **:** that is a, b, c, $B^2 - C^2 = 0$; $5A^2C^2 \neq 4(A^2 + C^2)$. With the center in the origin, particular relations concerning eccentricities must be satisfied.

In the second example we distribute the parametrization over the last two ellipsoids: eccentricity parameters referring to x, z and the y coordinate of the center are parametric for f_2 while the "dual" happens for f_3.

$$F = \begin{cases} f_1 = x^2 + y^2 + 4z^2 - 1 \\ f_2 = x^2C^2 + A^2C^2(y-b)^2 + A^2z^2 - A^2C^2 \\ f_3 = (x-a)^2B^2 + y^2 + (z-c)^2B^2 - B^2 \end{cases}$$

Here ${}^{\#}G_2^0 = 3$, and the first criterion sometimes applies. For $L_1 = [1,1,0,0]$ ($G_2^{L_1} = \langle xH, z^2, y^2, x^2 \rangle$) we have 4 terms, while for $L_2 = [0,1,1,0]$ (with $G_2^{L_2} = \langle z^2, x^2 \rangle$) we instead have just 2. There are also other lists satisfying the criterion.

The conclusion is that if φ is such that one of the following holds

$$3A^2B^2C^2 - A^2B^2 + B^2C^2 + A^2 - 4C^2 \ , \ B^2 - 1 \ , \ C(A^2 - 1) \neq 0 \tag{3}$$

$$B^2 - 1 = 0 \quad , \quad C(A^2 - 1) \neq 0 \tag{4}$$

$$C(A^2 - 1) = 0 \quad , \quad A^2 - 4C^2 \neq 0 \ , \ B^2 - 1 \neq 0 \tag{5}$$

$$4C^2 - 1 \ , \ B^2 - 1 \ , \ A^2 - 1 = 0 \quad , \quad b \neq 0 \tag{6}$$

$$\begin{gathered} bc \ , \ ab \ , \ b(B^2 - 4) \ , \ c(A^2 - 4C^2) \ , \ bA \ , \ a(A^2 - 4C^2) \ , \ cB(4C^2 - 1) = 0 \\ aB(4C^2 - 1) = 0 \ , \ 3A^2B^2C^2 - B^2(A^2 - C^2) + A^2 - 4C^2 = 0 \\ C(A^2 - 1) \ , \ B^2 - 1 \ , \ A^2B^2 - 3A^2C^2 - 4B^2C^2 - A^2 + 7C^2 \neq 0 \end{gathered} \tag{7}$$

$$\begin{gathered} a \ , \ b(B^2 - 4) \ , \ bA \ , \ 3A^2B^2C^2 - B^2(A^2 - C^2) + A^2 - 4C^2 = 0 \\ c(A^2 - 4C^2) \ , \ B^2 - 1 \ , \ C(A^2 - 1) \neq 0 \end{gathered} \tag{8}$$

then $\varphi(G)$ is a GB. It may be interesting to examine things deeply: here G is

$$\begin{aligned} g_1 = {} & 3A^2B^2C^2z^2 - A^2B^2z^2 + B^2C^2z^2 + A^2z^2 - 4C^2z^2 + 2aA^2B^2C^2x - 2aB^2C^2x + \\ & 2bA^2B^2C^2y - 2bA^2C^2y + 2cA^2B^2C^2z - 2cB^2C^2z - a^2A^2B^2C^2 - b^2A^2B^2C^2 - \\ & c^2A^2B^2C^2 + b^2A^2C^2 + a^2B^2C^2 + c^2B^2C^2 + A^2B^2C^2 - A^2C^2 - B^2C^2 + C^2 \\ g_2 = {} & B^2y^2 - y^2 + 3B^2z^2 + 2aB^2x + 2cB^2z - a^2B^2 - c^2B^2 \\ g_3 = {} & A^2C^2y^2 - C^2y^2 + A^2z^2 - 4C^2z^2 - 2bA^2C^2y + b^2A^2C^2 - A^2C^2 + C^2 \\ g_4 = {} & x^2 + y^2 + 4z^2 - 1 \end{aligned}$$

With values $(1, 2, 3, 2, -2, 1)$, satisfying condition 3, we obtain

$$\varphi(G) \colon \begin{cases} \varphi(g_1) : 3\,(12z^2 + 8x + 16y + 24z - 53) \\ \varphi(g_2) : 3y^2 + 12z^2 + 8x + 24z - 40 \\ \varphi(g_3) : 3y^2 - 16y + 13 \\ \varphi(g_4) : x^2 + y^2 + 4z^2 - 1 \end{cases} \qquad G' \colon \begin{cases} g_1' : 12z^2 + 8x + 16y + 24z - 53 \\ g_2' : 3y^2 - 16y + 13 \\ g_3' : x^2 + y^2 + 4z^2 - 1 \end{cases}$$

We can see that: $\varphi(G)$ is *not* reduced, G' has strictly less polynomials, and for G, $\varphi(G)$ and G' (as it should be) the $\mathbf{Z}$–monomial ideal is generated by $\{x^2, y^2, z^2\}$. With $(1, 2, 3, 2, 1, 1)$, values satisfying condition 4, we have

$$\varphi(G) : \begin{cases} \varphi(g_1) : 3\,(3z^2 + 2x + 6z - 10) \\ \varphi(g_2) : 3z^2 + 2x + 6z - 10 \\ \varphi(g_3) : 3y^2 - 16y + 13 \\ \varphi(g_4) : x^2 + y^2 + 4z^2 - 1 \end{cases} \qquad G' : \begin{cases} g_1' : 3z^2 + 2x + 6z - 10 \\ g_2' : 3y^2 - 16y + 13 \\ g_3' : 3x^2 - 8x + 16y - 24z + 24 \end{cases}$$

Specializations corresponding to the other conditions behave similarly.

A.3 The *Kruppa* System

The last example [10] is a system coming from a calibrating problem in vision with not exact coefficients of 6 quadratic equations in 5 unknowns $\{x_1, \ldots, x_5\}$ with a single solution. Being over-determined and approximated, practically all the efforts to obtain directly the solution using floating point GB fail. One can

1. solve five equations and substitute the found roots in the remaining one,
2. stop Buchberger's algorithm just before 1 appears in the basis and solve the linear system that is obtained,
3. use hybrid coefficients with an appropriate mod-p-driven zero test [19].

We propose a different approach, that seems a bit more systematic: substitute the constant term $v = 6.9769247755906706053$ in the first polynomial with a parameter a, to have 6 variables. Do the computation with floating point coefficients with 850 binary digits of precision (to overcome algorithmic error propagation) obtaining $G = \{p_0(a), x_1 - p_1(a), \ldots, x_5 - p_5(a)\}$ with $\deg(p_0) = 32$, $\deg(p_1) = \cdots = \deg(p_5) = 31$. It happens, as expected, that v is a (approximated) root of $p_0(a)$, and it is therefore possible to obtain the desired solution with a simple substitution $x_1 = p_1(v)\ , \ldots ,\ x_5 = p_5(v)$.

Investigation of the Stability Problem for the Critical Cases of the Newtonian Many-Body Problem

E.A. Grebenicov[1], D. Kozak-Skoworodkin[2], and M. Jakubiak[2]

[1] Computing Center of RAS, Moscow, Russia
[2] University of Podlasie, Poland

Using the computer algebra methods, different authors have proved the existence of new classes of the homografic solutions, in the Lagrange–Wintner sense [1], in the Newtonian many-body problem [2–6]. E.A. Grebenicov has also shown that any homographic solution of the n-body problem generates a new dynamical model, namely, "the restricted Newtonian $(n+1)$-body problem". These problems are similar to the famous "restricted three-body problem" which was proposed for the first time by K. Jacobi [7]. Then the theorems of the existence of stationary solutions (the equilibrium positions) for some fixed values of the parameter n were proved [8–10], and the problem of studying the stability of these solutions in the Lyapunov sense was formulated. The study of this problem can be done only on the basis of the KAM-theory [11–13] and only for the dynamical systems with two degrees of freedom it can be realized. We have shown that all the planar restricted n-body problems belong to this class for any n. The situation is essentially complicated for the critical (resonance) cases, when the eigenvalues of the linearized system of differential equations in the neighborhood of the stationary solution are rationally commensurable. In these critical cases the stability problem for hamiltonian dynamics may be studied only on the basis of Markeev and Sokolsky theorems [14,15]. These theorems contain mathematical estimations of the influence of so-called "non-annihilable resonance terms" in the Poincaré–Birkhoff normalizing transformations, which must be taken into account in the theorems on the stability of stationary solutions of hamiltonian equations in critical cases [16].

At first let us formulate the Arnold–Moser's theorem on the stability of hamiltonian systems with two degrees of freedom [14] and Markeev's theorem on the stability of stationary solutions in the critical (frequency resonances) cases.

Let us consider the fourth-order Hamiltonian system

$$\begin{cases} \frac{dp_k}{dt} = -\frac{\partial H}{\partial q_k}, \quad \frac{dq_k}{dt} = \frac{\partial H}{\partial p_k}, \\ k = 1, 2, \end{cases} \tag{1}$$

where the Hamiltonian $H(p_1, p_2, q_1, q_2)$ is an analytical function in the domain G_4 of 4-dimensional phase point

$$p_1 = p_2 = q_1 = q_2 = 0. \tag{2}$$

It means that the Hamiltonian H may be represented in G_4 as the following expansion:

$$H = H_2 + H_3 + H_4 + ..., \tag{3}$$

where H_m is the mth order homogeneous function with respect to p_1, p_2, q_1, q_2.

Besides, let the phase point (2) be a stationary solution of system (1). Then, replacing the Hamiltonian $H(p, q)$ in equations (1) with the function $H_2(p, q)$, we obtain the system

$$\begin{cases} \frac{dp_k}{dt} = -\frac{\partial H_2}{\partial q_k}, \quad \frac{dq_k}{dt} = \frac{\partial H_2}{\partial p_k}, \\ k = 1, 2, \end{cases} \tag{4}$$

which is just a linear system with constant matrix.

Let us assume also that, using the Birkhoff normalizing algorithm [16], we have succeeded in transformation of the Hamiltonian (3) in the domain G_4 into the normal form

$$\begin{aligned} &H(p_1, p_2, q_1, q_2) \equiv H^*(\tau_1, \tau_2, \varphi_1, \varphi_2) = \\ &= \sigma_1\tau_1 - \sigma_2\tau_2 + c_{20}\tau_1^2 + c_{11}\tau_1\tau_2 + c_{02}\tau_2^2 + H_5^*(\tau_1, \tau_2, \varphi_1, \varphi_2) + ..., \end{aligned} \tag{5}$$

where $\tau_1, \tau_2, \varphi_1, \varphi_2$ are new canonical variables and

$$\tau_k = \frac{p_k^2 + q_k^2}{2}, \quad k = 1, 2. \tag{6}$$

Note that the Birkhoff normal form (5) can be obtained only in the case when the eigenvalues of matrix standing in the right-hand side of system (4) are not resonant. Then the following Arnold–Moser theorem holds [14].

Theorem 1. *If the Hamiltonian (5) satisfies the following conditions:*
1) the eigenvalues of matrix (4) are purely imaginary numbers,
2) $k_1\sigma_1 + k_2\sigma_2 \neq 0$, where k_1, k_2 are integer numbers satisfying the inequality

$$0 < |k_1| + |k_2| \leq 4, \tag{7}$$

3)

$$c_{20}\sigma_2^2 + c_{11}\sigma_1\sigma_2 + c_{02}\sigma_1^2 \neq 0, \tag{8}$$

then the stationary solution (2) of system (1) is stable not only in linear approximation but in Lyapunov sense as well.

The condition 2) of the theorem just means that the third and the fourth order critical cases are absent [19].

Now let the frequencies of system (4) be connected by the third order resonance relation [19] $\sigma_1 = 2\sigma_2$. Assume that using the Birkhoff normalizing algorithm [16] we have transformed the Hamiltonian (3) in the neighborhood of phase point (2) into the form

$$\begin{aligned} H(p_1, p_2, q_1, q_2) \equiv H^*(\tau_1, \tau_2, \varphi_1, \varphi_2) = \\ = 2\sigma_2\tau_1 - \sigma_2\tau_2 + c_{20}\tau_1^2 + c_{11}\tau_1\tau_2 + c_{02}\tau_2^2 - \\ -\sqrt{\sigma_2(x_{1002}^2 + y_{1002}^2)}\tau_2\sqrt{\tau_1}\sin(\varphi_1 + 2\varphi_2) + H_5^*(\tau_1, \tau_2, \varphi_1, \varphi_2) + ..., \end{aligned} \tag{9}$$

where $\tau_1, \tau_2, \varphi_1, \varphi_2$ are new canonical variables. According to the Poincaré–Birkhoff theory of normal forms, the sixth term in expression (9), which corresponds to the critical case, is just their invariant and it is always present in the normal forms being similar to (9). The following theorem takes place [14].

Theorem 2. *If the inequality* $x_{1002}^2 + y_{1002}^2 \neq 0$ *in the Hamiltonian (9) is satisfied then the stationary solution (2) is unstable. But if* $x_{1002}^2 + y_{1002}^2 = 0$ *and* $c_{20} + 2c_{11} + 4c_{02} \neq 0$*, then it is stable in the Lyapunov sense.*

Let the frequencies of the system (4) be connected by the fourth-order resonance relation $\sigma_1 = 3\sigma_2$. Assume that using the Birkhoff normalizing algorithm [16] we have transformed Hamiltonian (3) in the neighborhood of phase point (2) into the form

$$\begin{gathered} H(p_1, p_2, q_1, q_2) \equiv H^*(\tau_1, \tau_2, \varphi_1, \varphi_2) = \\ = 3\sigma_2\tau_1 - \sigma_2\tau_2 + c_{20}\tau_1^2 + c_{11}\tau_1\tau_2 + c_{02}\tau_2^2 + \\ + \tfrac{1}{3}\sigma_2\sqrt{3(x_{1003}^2 + y_{1003}^2)}\tau_2\sqrt{\tau_1\tau_2}\cos(\varphi_1 + 3\varphi_2) + H_5^*(\tau_1, \tau_2, \varphi_1, \varphi_2) + ..., \end{gathered} \tag{10}$$

where $\tau_1, \tau_2, \varphi_1, \varphi_2$ are new canonical variables, and the sixth term in (10) is just the fourth-order resonance term. Let us denote

$$a = c_{20} + 3c_{11} + 9c_{02}, \quad b = 3\sigma_2\sqrt{(x_{1003}^2 + y_{1003}^2)}. \tag{11}$$

Then the following Markeev's theorem takes place [14].

Theorem 3. *If parameters (11) of the Hamiltonian (10) are such that* $\mid a \mid < b$ *then the equilibrium position (2) is unstable. But it is stable in the Lyapunov sense if* $\mid a \mid > b$.

Using the theorems above, we can investigate the stability of the equilibrium positions in new dynamical models for $n > 3$ bodies. Note that we can use theorems 1–3 only after transforming the Hamiltonian of the studied system to the Birkhoff normal form with accuracy up to the third or fourth order with respect to the local coordinates.

Let us now consider the stability problem for the equilibrium positions in the restricted problems of $n > 3$ bodies in the case of the fourth-order resonance. Let a point S with phase coordinates $(x^*, y^*, p_{x^*} = -\omega_n y^*, p_{y^*} = \omega_n x^*)$ be stable in linear approximation [8].

Angular velocity of the gravitational polygon $P_1 ... P_n$ attracting a body of infinitesimal mass is not arbitrary and is determined uniquely by the following general formula [8]:

$$\omega_n^2 = \frac{1}{a_0^3}\left[m_0 + \frac{m}{4}\sum_{k=2}^{n}\left(\sin\frac{\pi(k-1)}{n}\right)^{-1}\right], \tag{12}$$

a_0 is a radius of the circle circumscribed about the regular polygon,
m is a mass of every body $P_1, ..., P_n$ being in the vertices of the polygon,
m_0 is a mass of the body being in the center of the polygon.

Differential equations of the restricted many-body problem in the four-dimensional canonical phase space of local Lagrangian coordinates and momenta (X, Y, P_X, P_Y), (the origin of the local reference frame is in any equilibrium position $(x^*, y^*, p_{x^*}, p_{y^*})$, so $(X = x - x^*, Y = y - y^*, P_X = p_x - p_{x^*}, P_Y = p_y - p_{y^*})$) may be written in the canonical form

$$\begin{cases} \frac{dX}{dt} = \frac{\partial H}{\partial P_X}, & \frac{dP_X}{dt} = -\frac{\partial H}{\partial X}, \\ \frac{dY}{dt} = \frac{\partial H}{\partial P_Y}, & \frac{dP_Y}{dt} = -\frac{\partial H}{\partial Y}, \end{cases} \tag{13}$$

where the Hamiltonian is [19]

$$H(X, Y, P_X, P_Y) = \frac{1}{2}(P_x^2 + P_Y^2) + \omega_n(XP_Y - YP_X) - U(X, Y), \tag{14}$$

$$\begin{cases} U(X, Y) = f\left(\frac{m_0}{\Delta_0} + m\sum\limits_{i=1}^{n}\frac{1}{\Delta_i}\right), \\ \Delta_0 = \sqrt{X^2 + Y^2}, \\ \Delta_i = \sqrt{(X - X_i)^2 + (Y - Y_i)^2}, \quad i = 1, ..., n. \end{cases}$$

Differential equations (13) obviously have a particular solution (equilibrium point)

$$X = Y = P_X = P_Y = 0, \tag{15}$$

which corresponds to any of the stationary phase points being investigated.

Using vector notation we can rewrite the linear part of system (13) in the form

$$\left[\frac{d\tilde{X}}{dt}, \frac{d\tilde{Y}}{dt}, \frac{d\tilde{P}_X}{dt}, \frac{d\tilde{P}_Y}{dt}\right]^T = A \cdot \left[\tilde{X}, \tilde{Y}, \tilde{P}_X, \tilde{P}_Y\right]^T, \tag{16}$$

where A is a 4×4 matrix whose elements are obtained by means of differentiating the quadratic terms in the expansion of the Hamiltonian (14) in the neighborhood of point (15). Let us denote the absolute values of the eigenvalues of the matrix A as

$$|\lambda_1| = |\lambda_2| = \sigma_1, \quad |\lambda_3| = |\lambda_4| = \sigma_2. \tag{17}$$

They are equal in pairs because the matrix A is symplectic. Assume that we have the critical case $\sigma_1 = 3\sigma_2$.

As the Hamiltonian $H(X, Y, P_X, P_Y)$ in (14) is an analytic function in the neighborhood of the point $(0, 0, 0, 0)$, then with the help of the computer algebra system "Mathematica" [17] we can easily find its representation in the form (3) as

$$H = H_2(X, Y, P_X, P_Y) + H_3(X, Y) + H_4(X, Y) + ..., \tag{18}$$

where $H_k (k = 2, 3, ...)$ is the kth order homogeneous form of phase coordinates, in particular,

$$H_2 = \frac{1}{2}a_{20}X^2 + \frac{1}{2}a_{02}Y^2 + \frac{1}{2}(P_X^2 + P_Y^2) + a_{11}XY + \omega_n(YP_X - XP_Y), \tag{19}$$

$$H_3 = a_{30}X^3 + a_{03}Y^3 + a_{21}X^2Y + a_{12}XY^2, \tag{20}$$

$$H_4 = a_{40}X^4 + a_{04}Y^4 + a_{31}X^3Y + a_{13}XY^3 + a_{22}X^2Y^2. \tag{21}$$

As the second-order expression $H_2(X, Y, P_X, P_Y)$ defined in (19) contains the term $\omega_n(YP_X - XP_Y)$ which is not a positive definite quadratic form then the Hamiltonian H has no properties of the Lyapunov function [18]. Hence, the first problem is to construct such a linear non-degenerate canonical transformation which reduces a quadratic part of the Hamiltonian to the Birkhoff normal form [16]

$$H_2 \equiv K_2 = \frac{1}{2}(p_1^2 + \sigma_1^2 q_1^2) - \frac{1}{2}(p_2^2 + \sigma_2^2 q_2^2), \tag{22}$$

where the products of coordinates and momenta of the form $(q_1p_1, q_1p_2, q_1q_2, q_2p_1, q_2p_2, p_1p_2)$ are absent. We shall look for such a linear transformation $(X, Y, P_X, P_Y) \to (q_1, q_2, p_1, p_2)$ in the form

$$\begin{bmatrix} X \\ Y \\ P_X \\ P_Y \end{bmatrix} = B_4 \cdot \begin{bmatrix} q_1 \\ q_2 \\ p_1 \\ p_2 \end{bmatrix}, \tag{23}$$

where B_4 is an unknown 4×4 transformation matrix. According to the theory of canonical transformations it is known that the matrix B_4 must be symplectic, i.e., its elements must satisfy the following identity:

$$B_4^T I_4 B_4 = I_4, \quad I_4 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{24}$$

where B^T is a matrix transposed with respect to B.

After some necessary matrix transformations given in [8], the entries b_{ij} $(i, j = 1, 2, 3, 4)$ of matrix B_4 are determined as solutions of the following system of linear homogeneous algebraic equations:

$$C_{16} \cdot z = 0, \tag{25}$$

where z is the 16-dimensional vector whose components are just the elements of the matrix $B_4 : z^T = (b_{11}, b_{12}, ..., b_{44})$. C_{16} is the 16×16 matrix of the form

$$\begin{vmatrix}
0 & 0 & \sigma_1^2 & 0 & \omega_n & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\sigma_2^2 & 0 & \omega_n & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & \omega_n & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & \omega_n & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-\omega_n & 0 & 0 & 0 & 0 & 0 & \sigma_1^2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & -\omega_n & 0 & 0 & 0 & 0 & 0 & -\sigma_2^2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & -\omega_n & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & -\omega_n & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
-a_{20} & 0 & 0 & 0 & -a_{11} & 0 & 0 & 0 & 0 & 0 & \sigma_1^2 & 0 & \omega_n & 0 & 0 & 0 \\
0 & -a_{20} & 0 & 0 & 0 & -a_{11} & 0 & 0 & 0 & 0 & 0 & -\sigma_2^2 & 0 & \omega_n & 0 & 0 \\
0 & 0 & -a_{20} & 0 & 0 & 0 & -a_{11} & 0 & -1 & 0 & 0 & 0 & 0 & 0 & \omega_n & 0 \\
0 & 0 & 0 & -a_{20} & 0 & 0 & 0 & -a_{11} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \omega_n \\
-a_{11} & 0 & 0 & 0 & -a_{02} & 0 & 0 & 0 & -\omega_n & 0 & 0 & 0 & 0 & 0 & \sigma_1^2 & 0 \\
0 & -a_{11} & 0 & 0 & 0 & -a_{02} & 0 & 0 & 0 & -\omega_n & 0 & 0 & 0 & 0 & 0 & -\sigma_2^2 \\
0 & 0 & -a_{11} & 0 & 0 & 0 & -a_{02} & 0 & 0 & 0 & -\omega_n & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & -a_{11} & 0 & 0 & 0 & -a_{02} & 0 & 0 & 0 & -\omega_n & 0 & 1 & 0 & 0
\end{vmatrix}$$

The calculations have shown that determinant of the matrix C_{16} and all its fifteenth, fourteenth, and thirteenth order minors are equal to zero. But there are some nonzero minors of the twelfth order. Hence, the algebraic system (23) has nonzero solutions, and these solutions form a four-parametric family.

Four elements of the matrix B_4 must be chosen in such a way that quadratic form of the transformed Hamiltonian has the Birkhoff normal form (22). It means that among the infinite set of matrices B_4 we have to choose a symplectic matrix, i.e., it must be such a solution of the system (23) which satisfies the identity (24).

Doing a transformation (23) of functions (19)–(21) with matrix B_4, we obtain a new Hamiltonian in the form

$$\begin{aligned} K(q_1,q_2,p_1,p_2) &= K_2+K_3+K_4+\ldots = \tfrac{1}{2}(p_1^2+\sigma_1^2q_1^2)-\tfrac{1}{2}(p_2^2+\sigma_2^2q_1^2)+\\ &+\sum_{\nu_1+\nu_2+\mu_1+\mu_2=3} k_{\nu_1\nu_2\mu_1\mu_2}q_1^{\nu_1}q_2^{\nu_2}p_1^{\mu_1}p_2^{\mu_2}+\\ &+\sum_{\nu_1+\nu_2+\mu_1+\mu_2=4} k_{\nu_1\nu_2\mu_1\mu_2}q_1^{\nu_1}q_2^{\nu_2}p_1^{\mu_1}p_2^{\mu_2}+\ldots. \end{aligned} \tag{26}$$

Thus, we have realized the first step which is necessary for investigation of the Lyapunov stability of the equilibrium positions in the critical case $\sigma_1 = 3\sigma_2$.

In order to reduce the new Hamiltonian $K(q_1,q_2,p_1,p_2) = K_2+K_3+K_4+\ldots$ to the form which is convenient for applying the second Birkhoff transformation, let us do the following canonical transformation [14]:

$$\begin{cases} q_1 = \frac{1}{2}q_{11}+\frac{i}{\sigma_1}p_{11}, & q_2 = -\frac{i}{2}q_{22}+\frac{1}{\sigma_2}p_{22},\\ p_1 = \frac{1}{2}i\sigma_1 q_{11}+p_{11}, & p_2 = -\frac{1}{2}\sigma_2 q_{22}+ip_{22} \end{cases} \quad (i^2=-1). \tag{27}$$

As result the Hamiltonian $K(q_1,q_2,p_1,p_2)$ takes the form

$$\begin{aligned} F(q_{11},q_{22},p_{11},p_{22}) &= F_2+F_3+F_4+\ldots = i\sigma_1q_{11}p_{11}+i\sigma_2q_{22}p_{22}+\\ &+\sum_{\nu_1+\nu_2+\mu_1+\mu_2=3} f_{\nu_1\nu_2\mu_1\mu_2}q_{11}^{\nu_1}q_{22}^{\nu_2}p_{11}^{\mu_1}p_{22}^{\mu_2}+\\ &+\sum_{\nu_1+\nu_2+\mu_1+\mu_2=4} f_{\nu_1\nu_2\mu_1\mu_2}q_{11}^{\nu_1}q_{22}^{\nu_2}p_{11}^{\mu_1}p_{22}^{\mu_2}+\ldots. \end{aligned} \tag{28}$$

Here we use the Markeev's notation [14]: $f_{v_1v_2\mu_1\mu_2} = x_{v_1v_2\mu_1\mu_2}+iy_{v_1v_2\mu_1\mu_2}$.

Applying now one more Birkhoff transformation which was written by A.P. Markeev in a very convenient form, we cancel all the third-order terms in the Hamiltonian $F(q_{11},q_{22},p_{11},p_{22})$ and reduce the fourth-order terms to the following form [14]:

$$\begin{aligned} W_4(Q_1,Q_2,P_1,P_2) &= -c_{20}Q_1^2P_1^2+c_{11}Q_1Q_2P_1P_2-c_{02}Q_2^2P_2^2+\\ &+(x_{1003}+iy_{1003})Q_1P_2^3+\left(-\tfrac{\sigma_2^2}{12}(x_{1003}-iy_{1003})\right)Q_2^3P_1. \end{aligned} \tag{29}$$

where

$$\begin{aligned} c_{20} &= -f_{2020}-\tfrac{27}{8}\sigma_2^2(x_{0030}^2+y_{0030}^2)-\tfrac{3}{2}(x_{1020}^2+y_{1020}^2)-\tfrac{9}{10}(x_{0120}^2+y_{0120}^2)+\\ &+\tfrac{1}{2}(x_{1011}^2+y_{1011}^2)+\tfrac{9}{56}\sigma_2^2(x_{0021}^2+y_{0021}^2), \end{aligned}$$

$$\begin{aligned} c_{11} &= f_{1111}-\tfrac{2}{3}(x_{1002}^2+y_{1002}^2)+\tfrac{3}{10}\sigma_2^2(x_{0012}^2+y_{0012}^2)-\tfrac{9}{14}\sigma_2^2(x_{0021}^2+y_{0021}^2)-\\ &-\tfrac{18}{5}(x_{0120}^2+y_{0120}^2)+2(x_{0111}x_{1020}+y_{0111}y_{1020})-\\ &-\tfrac{4}{\sigma_2}(x_{0201}y_{1011}+x_{1011}y_{0201}), \end{aligned}$$

$$c_{02} = -f_{0202} + \tfrac{3}{8}\sigma_2^2(x_{0003}^2 + y_{0003}^2) + \tfrac{6}{\sigma_2^2}(x_{0201}^2 + y_{0201}^2) - \tfrac{1}{6}(x_{1002}^2 + y_{1002}^2) - \\ -\tfrac{1}{2}(x_{0111}^2 + y_{0111}^2) - \tfrac{3}{40}\sigma_2^2(x_{0012}^2 + y_{0012}^2),$$

$$x_{1003} = u_{1003} - \tfrac{9}{5}(x_{0120}x_{0012} + y_{0120}y_{0012}) - \tfrac{1}{\sigma_2}(x_{1002}y_{1011} + x_{1011}y_{1002}) + \\ +\tfrac{4}{\sigma_2^2}(x_{1002}x_{0201} + y_{1002}y_{0201}) + \tfrac{3}{2}(x_{0003}x_{0111} + y_{0003}y_{0111}),$$

$$y_{1003} = v_{1003} - \tfrac{9}{5}(x_{0120}y_{0012} - x_{0012}y_{0120}) - \tfrac{1}{\sigma_2}(y_{1011}y_{1002} - x_{1011}x_{1002}) + \\ +\tfrac{4}{\sigma_2^2}(x_{0201}y_{1002} - x_{1002}y_{0201}) + \tfrac{3}{2}(x_{0111}y_{0003} - x_{0003}y_{0111}),$$

and

$$u_{1003} = \frac{1}{2}\sigma_1 k_{0013} + \frac{1}{2\sigma_2^3}k_{1300} - \frac{1}{2\sigma_2}k_{1102} - \frac{\sigma_1}{2\sigma_2^2}k_{0211},$$

$$v_{1003} = -\frac{\sigma_1}{2\sigma_2}k_{0112} - \frac{1}{2}k_{1300} + \frac{1}{2\sigma_2^2}k_{1201} + \frac{\sigma_1}{2\sigma_2^3}k_{0310}.$$

Now we can calculate the parameters

$$a = c_{20} + 3c_{11} + 9c_{02}, \quad b = 3\sigma_2\sqrt{(x_{1003}^2 + y_{1003}^2)},$$

which have been introduced in Theorem 3. Comparing the values of $\mid a \mid$ and b we can answer the question whether the equilibrium position $(0, 0, 0, 0)$ is stable or unstable in the Lyapunov sense. This may be demonstrated in the best way for some fixed value of the main dynamical parameter m.

Example.
For the point S with the phase coordinates

$$x^* = 0.809543..., \quad y^* = 0.588167...,$$

which have been calculated for $n = 5$ and $m = 0.0017525812892316 07...$ (i.e., in the restricted problem of seven bodies in the case of the fourth-order resonance ($\sigma_1 = 0.944354...$)), the matrix B_4 has the form [19]:

$$B_4 = \begin{bmatrix} 0 & 0 & 1.65822... & 3.82824... \\ -1.52333... & 0.659837... & -1.20056... & -4.68026... \\ 0.0463528... & -0.281294... & 1.20201... & 4.68591... \\ 1.07067... & -0.463764... & 0.136895... & 3.17302... \end{bmatrix}.$$

Doing all the calculations, as a result we obtain the Hamiltonian in the form

$$\tilde{H}(T_1, T_2, \varphi_1, \varphi_2) = 3\sigma_2 T_1 - \sigma_2 T_2 - 1.11118T_1^2 - 11.9648T_1T_2 - 7.88667T_2^2 + \\ +34.4172\tfrac{\sqrt{3}}{3}\sigma_2 T_2\sqrt{T_1T_2}\cos(\varphi_1 + 3\varphi_2) + ...,$$

which is necessary for using the main Theorem 3 formulated at the beginning of the present paper, and the parameters a and b are given by

$$\mid a \mid = \mid c_{20} + 3c_{11} + 9c_{02} \mid = 107.986, \quad b = 3\sigma_2 \cdot 34.4172 = 32.502.$$

Comparing the values of $\mid a \mid$ and b, we see that $\mid a \mid > b$. Hence, the equilibrium position S is stable in the Lyapunov sense in the case of the fourth-order resonance.

References

1. Wintner, A.: The Analytical Foundations of Celestial Mechanics, Princeton Univ. Press, Princeton (1941)
2. Elmabsout, B.: Sur l'existence de certaines configurations d'équillibre rélatif dans le problème des n corps. Celestial Mech. and Dynamical Astron. **4** No. **1** (1988) 131–151
3. Grebenicov, E.A.: The existence of the exact symmetrical solutions in the planar Newtonian many-body problem (in Russian). Mathematical Modeling **10**, No. **8** (1998) 74–80.
4. Zemtsova, N.I.: Stability of the stationary solutions of the differential equations of restricted Newtonian problem with incomplete symmetry.In: Nonlinear Dynamics and Systems Theory, Kiev **3(1)** (2003) 105–116
5. Bang, D., Elmabsout, B.: Configurations polygonales en equilibre relative. C.R. Acad. Sci., Série Iib. **329** (2001) 243–248
6. Grebenicov, E.A., Prokopenya, A.N.: On the existence of a new class of the exact solutions in the planar Newtonian many-body problem (in Russian). In: The Questions of Modeling and Analysis in the Problems of Making Decision, V.A. Bereznev (Ed.), Computing Center RAS, Moscow (2004) 39–57
7. Jacobi, K.: Gesammelte Werke, Bd. **1-7**, Berlin (1881–1891)
8. Grebenicov, E.A., Kozak-Skoworodkin, D., Jakubiak, M.: Methodes of Computer Algebra in Many-Body Problem (in Russian), Ed. of UFP, Moscow (2002)
9. Siluszyk, A.: Problem on linear stability of stationary solutions of restricted 8-body problem with incomplete symmetry (in Russian), BRDU, Brest, No. **2** (2004) 20–26
10. Ikhsanov, E.V.: Stabilty of equilibrium state in restricted 10-body problem for resonance case of 4^{th} order. In: The Questions of Modeling and Analysis in the Problems of Making Decision (in Russian), V.A.Bereznev (Ed.), Computing Center RAS, Moscow (2004) 16–23
11. Kolmogorov, A.N.: General theory of dynamical systems and classical mechanics (in Russian). In: Int. Math. Congress in Amsterdam, Physmatgiz, Moscow (1961) 187–208
12. Arnold, V.I.: About stability of equilibrium positions of Hamiltonian systems in general eliptic case (in Russian). DAN USSR **137** (1961) 255–257
13. Moser, J.K.: Lectures on Hamiltonian Systems. Courant Institute of Mathematical Science, New York (1968) 295
14. Markeev, A.P.: Libration Points in Celestial Mechanics and Cosmodynamics (in Russian), Nauka, Moscow (1974)
15. Sokol'sky, A.G.: About stability of Hamiltonian autonomy system with the resonance of first order (in Russian). J. Appl. Math. and Mech. **41** (1977) 24–33
16. Birkhoff, G.D.: Dynamical Systems (in Russian), GITTL, Moscow (1941)
17. Wolfram, S.: The Mathematica – Book, Cambridge University Press, Cambridge (1996)
18. Lyapunov, A.M.: General Problem on Stability of Motion (in Russian), Academy of Sciences of the USSR, Moscow **1** (1954)
19. Kozak-Skoworodkin, D.: The Stability of Equilibrium Points In Case of Resonance $\sigma_1 = 2\sigma_2$ in the Restricted Seven-Body Problem (in Russian), BRDU, Brest, No. **1(38)** (2004) 15–22

Symbolic-Numerical Algorithm for Solving the Time-Dependent Schrödinger Equation by Split-Operator Method

Alexander Gusev[1], Vladimir Gerdt[1], Michail Kaschiev[2], Vitaly Rostovtsev[1], Valentin Samoylov[1], Tatyana Tupikova[1], Yoshio Uwano[3], and Sergue Vinitsky[1]

[1] Joint Institute for Nuclear Research, Dubna, Moscow Region, Russia
vinitsky@thsun1.jinr.ru
[2] Institute of Mathematics and Informatics, BAS, Sofia, Bulgaria
[3] Future University-Hakodate, Hakodate, Japan

Abstract. A new computational approach is proposed for the solution of the time-dependent Schrödinger equation (TDSE), in which a symbolic algorithm named GATEO and a numerical scheme based on the finite-element method (FEM) are effectively composed. The GATEO generates the multi-layer operator-difference scheme for TDSE and evaluates the effective Hamiltonian from the original time-dependent Hamiltonian by means of the Magnus expansion and the Pade-approximation. In order to solve the TDSE with the effective Hamiltonian thus obtained, the FEM is applied to a discretization of spatial domain which brings the difference scheme in operator form to the one in algebraic form. The efficiency and accuracy of GATEO and the numerical scheme associated with FEM is confirmed in the second-, fourth-, and sixth-order time-step computations for certain integrable atomic models with external fields.

1 Introduction

The modern laser physics experiments have stimulated computer simulations for the time-dependent dynamics of few-body Coulomb systems (including exotic ones) in a train of laser pulses [1] and the time-dependent Schrödinger equation (TSDE) for the control problems of quantum systems [2]. For such subjects, the unitary splitting algorithms have been developed, for example, [3,4].

For any numerical method, a pair of requirements are always made: one is stability, and the other is accuracy. From the viewpoint of these requirements, the unitary splitting methods have a big advantage: the unitarity of the evolution operators applied in the methods preserves the norm of the wave functions, so that the conservation of probability density and robustness of the methods are guaranteed. In spite of the advantage due to the unitary preserving property, the unitary splitting methods still have had the following problem to be settled: At each time step associated with the splitting, a solution with a certain accuracy has to be determined. As a conventional method to this problem, the expansion of the wave packet of TDSE in a globally defined basis has been considered.

Instead of this expansion method, the finite-element method (FEM) [5,6] is applied together with using suitable interpolation technique connecting solutions given on neighboring spatial grids [7,8].

In this paper, a new computational method is proposed to solve the TDSE, in which the unitary splitting algorithm with uniform time grids [9] is combined with an application of FEM and an interpolation method. The second-, fourth-, and sixth-order approximations with respect to the time step are derived for a numerical computation. As for the spatial step, the FEM is applied to construct the numerical schemes with a required accuracy with respect to the spatial step [7,8], in which a special gauge transformation of effective Hamiltonian is fixed to ensure a high applicability of the FEM. The efficiency and accuracy of the developed numerical algorithms is confirmed in certain integrable atomic models in external fields.

The organization of this paper is outlined as follows. In section 2, a general formulation and the operator-difference multi-layer scheme for solving TDSE in a finite time interval are given. The algorithm GATEO is presented in the conventional pseudo-code and is implemented in Maple package, which evaluates the effective operators for the given Hamiltonian and the weights of the scheme with a required order of the accuracy. In section 3, unitary schemes for partial splitting of evolution operator are proposed that provide a symmetric matrix representation in the local support functions of the FEM. In section 4, a background of the algorithm for generating matrix problem in framework of the FEM is presented. In section 5, stability and efficiency of the developed schemes and algorithms are confirmed in certain integrable atomic models in external fields. Section 6 is devoted to concluding remarks and perspectives of further studies.

2 General Formulation and Calculation Schemes

Let us consider the Cauchy problem (the initial-value problem),

$$i\frac{\partial\psi(x,t)}{\partial t} = H(x,t)\psi(x,t), \quad ||\psi||^2 = \int |\psi(x,t)|^2 dx = 1, \tag{1}$$

$$\psi(x,t_0) = \psi_0(x), \quad \psi(x,t) \in \mathbf{H}^1(\mathbf{R}^n \otimes [t_0,T]), \quad \psi_0(x) \in \mathbf{H}^1(\mathbf{R}^n),$$

of the multi-dimensional TDSE on the time interval $t \in [t_0, T]$ with initial state $\psi_0(x)$, which describes an atomic model in an external (electromagnetic) field [1].

We rewrite (1) for an unitary operator $U(t, t_0, \lambda)$ carrying the initial state $\psi(x, t_0)$ to the solution $\psi(x, t)$ in the form

$$i\frac{\partial U(t,t_0,\lambda)}{\partial t} = \lambda H(x,t)U(t,t_0,\lambda), \quad U(t_0,t_0,\lambda) = 1,$$

where λ is the formal parameter for an expansion given below.

Let us consider the uniform grid,

$$\Omega_\tau[t_0,T] = \{t_0, t_{k+1} = t_k + \tau, (k = 0,1,...,K), t_K = T\} \tag{2}$$

with time step τ, in the time interval $[0, T]$.

[1] The atomic units are applied throughout this paper.

Table 1. Range of the summation indexes $(n, q, l_1, ..., l_q, j)$ at step **1:** of GATEO

n	q	l_1	l_2	l_3	j	n	q	l_1	l_2	l_3	l_4	j	n	q	l_1	l_2	l_3	l_4	l_5	j	n	q	l_1	l_2	l_3	l_4	j
1	0				1	1	1	4				5	1	1	5					6	2	1	4				6
1	1	1			2	1	2	3	1			5	1	2	4	1				6	2	2	3	1			6
2	0				2	1	2	2	2			5	1	2	3	2				6	2	2	2	2			6
1	1	2			3	1	2	1	3			5	1	2	2	3				6	2	2	1	3			6
1	2	1	1		3	1	3	2	1	1		5	1	2	1	4				6	2	3	2	1	1		6
2	1	1			3	1	3	1	2	1		5	1	3	3	1	1			6	2	3	1	2	1		6
3	0				3	1	3	1	1	2		5	1	3	2	2	1			6	2	3	1	1	2		6
1	1	3			4	1	4	1	1	1	1	5	1	3	1	3	1			6	2	4	1	1	1	1	6
1	2	2	1		4	2	1	3				5	1	3	2	1	2			6	3	1	3				6
1	2	1	2		4	2	2	2	1			5	1	3	1	2	2			6	3	2	2	1			6
1	3	1	1	1	4	2	2	1	2			5	1	3	1	1	3			6	3	2	1	2			6
2	1	2			4	2	3	1	1	1		5	1	4	2	1	1	1		6	3	3	1	1	1		6
2	2	1	1		4	3	1	2				5	1	4	1	2	1	1		6	4	1	2				6
3	1	1			4	3	2	1	1			5	1	4	1	1	2	1		6	4	2	1	1			6
4	0				4	4	1	1				5	1	4	1	1	1	2		6	5	1	1				6
						5	0					5	1	5	1	1	1	1	1	6	6	0					6

We express the unitary operator $U(t_{k+1}, t_k, \lambda)$ carrying the solution $\psi(t_k) \equiv \psi(x, t_k)$ at $t = t_k$ to the one $\psi(t_{k+1})$ at $t = t_{k+1}$ in the form [9]

$$\psi(t_{k+1}) = U(t_{k+1}, t_k, \lambda)\psi(t_k), \quad U(t_{k+1}, t_k, \lambda) = \exp\{-\imath\tau A_k(t, \lambda)\}. \tag{3}$$

Before going to the pseudo-code representation of the algorithm GATEO (Generation of Approximations of the Time-Evolution Operator) for the operator-difference scheme, we present what is be made in GATEO in a theoretical way.

Step 1: We start with the power-series expansion,

$$A_k(t, \lambda) = \frac{1}{\tau}\sum_{j=1}^{\infty} \lambda^j A_{(j)k}(t), \tag{4}$$

of $A_k(t, \lambda)$ in terms of the formal parameter [2] λ, where the operator-valued coefficients $A_{(j)}(t) \equiv A_{(j)k}(t)$ are evaluated from the operator-identity [10]

$$-\imath\lambda H(t) = \sum_{n=1}^{\infty}\sum_{q=0}^{\infty}\sum_{l_1,\ldots,l_q=1}^{\infty} \frac{\lambda^{n+\Sigma_{i=1}^{q} l_i}[A_{(l_1)}(t), [\ldots, [A_{(l_q)}(t), \dot{A}_{(n)}(t)]\ldots]]}{(q+1)!}, \tag{5}$$

where the ranges of the summation indices in (5) are given in Table 1. Note that the dot over the operator $A_{(n)}(t)$ means the partial derivation, $\dot{A}_{(n)}(t) = \partial_t A_{(n)}(t)$, in t.

[2] The λ will be replaced to be $\lambda = 1$ later.

Steps 2-3: Equating the coefficients at the same powers of λ in both sides of (5), we obtain a set of the first-order differential equations. Solving sequentially the set of equations thus obtained, we are led to the effective Hamiltonian $A_k(t) \equiv A_k(t, \lambda = 1)$ connected with the original one $H(t)$ via the Magnus expansion [10]

$$A_k(t_{k+1}) = \frac{1}{\tau}\int_{t_k}^{t_{k+1}} dt H(t) + \frac{\imath}{2\tau}\int_{t_k}^{t_{k+1}} dt' \int_{t_k}^{t'} dt''[H(t''), H(t')] + \cdots . \quad (6)$$

Step 4: Under a sufficient smoothness of $H(t)$ in t, we can obtain the unitary scheme,

$$U(t_{k+1}, t_k; \tau) = \exp(-i\tau A_k^{(M)}) + O(\tau^{2M+1}), \quad A_k^{(M)} \equiv A_k^{(M)}(t_{k+1}, \lambda = 1) \quad (7)$$

with truncation error $O(\tau^{2M})$, where $A_k^{(M)}(t, \lambda)$ is the abbreviation of the truncation,

$$A_k^{(M)}(t, \lambda) = \frac{1}{\tau}\sum_{j=1}^{M} \lambda^j A_{(j)k}(t), \quad (8)$$

of the expansion (4).

Steps 5-6: We wish to express the truncation $A_k^{(M)}$ given by (8) in terms of $H(t)$, its partial derivative in time and the higher ones. Putting the Taylor expansion of $H(t)$ made in a vicinity of $t = t_k + \tau/2$ into the integrals in (6), one can find an analytical (meaning non-numerical) expression of operators, $A_k^{(1)}, A_k^{(2)}, ..., A_k^{(M)}$, in principle: For $A_k^{(1)}$, we have only to calculate the coefficient of λ^0, and then obtain $A_k^{(1)} = \int_0^1 d\xi_0 H(t_k + \xi_0\tau) = H(t_k + \tau/2) + O(\tau^2)$, without any difficulties. However, in the case of $A_k^{(M)}$ with large M, rather cumbersome calculations are required to fix all the coefficients of the power of λ in $A_k^{(M)}$ "by hand". Our algorithm GATEO (Generation of Approximations of the Time-Evolution Operator) is thereby motivated by the difficulty of calculation pointed out above, that provides the set of the generators $A_k^{(M)}(t_{k+1})$ required in the operator-difference scheme derived in (15). To show the complexity of calculations, we present the first three approximations of Eq. (7) obtained after applying GATEO implemented in MAPLE package:

$$A_k^{(1)} = H, \quad (9)$$

$$A_k^{(2)} = A_k^{(1)} + \tau^2\left(\frac{1}{24}\ddot{H} - \frac{\imath}{12}[\dot{H}, H]\right), \quad (10)$$

$$A_k^{(3)} = A_k^{(2)} + \tau^4\left(\frac{1}{1920}\overset{....}{H} - \frac{1}{720}[[\ddot{H}, H], H] - \frac{1}{240}[\dot{H}, [\dot{H}, H]]\right.$$
$$\left. + \frac{\imath}{480}[\ddot{H}, \dot{H}] - \frac{\imath}{480}[\dddot{H}, H] - \frac{\imath}{720}[[[\dot{H}, H], H], H]\right), \quad (11)$$

where $H \equiv H(t_{k+1/2})$, $\dot{H} \equiv \partial_t H(x,t)|_{t=t_{k+1/2}}$, $\ddot{H} \equiv \partial_t^2 H(x,t)|_{t=t_{k+1/2}}$, $\ldots$, and $[\cdot, \cdot]$ indicates the commutator. In the pseudo-code representation given subse-

Table 2. Real and imaginary parts of coefficients $\alpha_\zeta^{(M)}$, $M = 1, 2, 3$, $\zeta = 1, ..., M$

M	ζ	$\Re\alpha_\zeta$	$\Im\alpha_\zeta$
1	1	0	−1.0
2	1	−0.57735026918962576450914878050	−1.0
2	2	+0.57735026918962576450914878050	−1.0
3	1	−0.81479955424892281841473623156	−0.85405673065166346526579940886
3	2	+0.0	−1.29188653869667306946840118228
3	3	+0.81479955424892281841473623156	−0.85405673065166346526579940886

quently, expressions, $t_k = t_{in}$, $t_{k+1/2} = t_k + \tau/2 = t_c$ and $t_{k+1} = t_k + \tau = t_{out}$, are adopted.

Steps 7-8: We wish to make further approximation of the unitary scheme (7). Application to the exponential operator on the rhs of (7) of the generalized $[M/M]$ Padé approximation yields

$$\exp(-i\tau A_k^{(M)}) = \prod_{\zeta=1}^{M} T_{\zeta k} + O(\tau^{2M+1}), \quad T_{\zeta k} = \frac{I + \frac{\tau}{2M}\alpha_\zeta^{(M)} A_k^{(M)}}{I + \frac{\tau}{2M}\overline{\alpha}_\zeta^{(M)} A_k^{(M)}}, \tag{12}$$

where the overline indicates the complex conjugate operation. The coefficients, $\alpha_\zeta^{(M)}$ ($\zeta = 1, \ldots, M$, $M \geq 1$), stand for the roots of the polynomial equation, ${}_1F_1(-M, -2M, 2M\imath/\alpha) = 0$, where ${}_1F_1$ is the confluent hypergeometric function. Table 2 lists the values of the coefficients, $\alpha_\zeta^{(M)}$ for $M = 1, 2, 3$, calculated at **step 7** in GATEO. The coefficients $\alpha_\zeta^{(M)}$ have the following properties: $\Im\alpha_\zeta^{(M)} < 0$ and $0.6 < |\alpha_\zeta^{(M)}| < \mu^{-1}$, where $\mu \approx 0.28$ is the root of equation $\mu \exp(\mu + 1) = 1$. Note that the condition $\tau < 2M\mu||A_k^{(M)}||^{-1}$ guarantees the validity of the approximation (12) for any bounded operator $A_k^{(M)}$.

We are now in a position to obtain the transition from $\psi(t_k)$ to $\psi(t_{k+1})$, by using the approximation (12) of the evolution operator in (3). To make it, we rewrite the transition in terms of the auxiliary functions defined by

$$\psi^{k+\zeta/M} = T_{\zeta k}\psi^{k+(\zeta-1)/M}, \quad \zeta = 1, ..., M. \tag{13}$$

The fact that $\Im\alpha_\zeta^{(M)} < 0$ yields the operators, $T_{\zeta k}$, to be isometric, so that all the $\|\psi^{k+\zeta/M}\|$ have an equal norm;

$$\|\psi^k\| = \|\psi^{k+1/M}\| = \ldots = \|\psi^{k+1}\|. \tag{14}$$

Putting (13) at each kth time step of the grid Ω_τ into an approximate solution of the TDSE (1), we are led to the operator-difference scheme [9],

$$\begin{array}{rcl} \psi^0 &=& \psi(t_0), \\ \left(I + \dfrac{\tau}{2M}\overline{\alpha}_\zeta^{(M)} A_k^{(M)}\right)\psi^{k+\zeta/M} &=& \left(I + \dfrac{\tau}{2M}\alpha_\zeta^{(M)} A_k^{(M)}\right)\psi^{k+(\zeta-1)/M}, \\ \zeta = 1, 2, \ldots, M & \text{and} & k = 0, 1, \ldots, K-1, \\ \psi(T) &=& \psi^K. \end{array} \tag{15}$$

Hence, the auxiliary functions $\psi^{k+\zeta/M}$ ($\zeta = 1, ..., M-1$) in Eq. (15) can be treated as a kind of approximate solutions on a set of the fractional time steps $t_{k+\zeta/M} = t_k + \tau\zeta/M$, $\zeta = 1, ..., M-1$ in the time interval $[t_k, t_{k+1}]$. The scheme (15) is an implicit one of order $2M$ preserving the norm of the difference solution(14), so that this scheme is stable. Further, the scheme (15) provides an approximation of the order $O(\tau^{2M})$ in the sense of [11], while any individual equation in (15) only provides an approximation of degree not higher than $O(\tau^2)$. Note that in the case $M = 1$, i.e., [1/1] Padé approximation of exponential operator (12), the scheme (15) reduces to the well-known Crank-Nicholson scheme [12].

Algorithm GATEO(Generation of Approximations of the TEO)

Input:
M is an order of approximation of the TEO with respect to the time variable;
t is the time variable;
$\tau = t_{out} - t_{in}$ is a step of the uniform grid $\Omega_\tau[t_0, T]$ with respect to t;
t_{in}, and t_{out} are boundary points of time interval $t \in [t_{in}, t_{out} = t_{in} + \tau]$;
$H(t) \equiv H(x,t)$ is the Hamiltonian;
$\psi(t_{in})$ is a wave packet at time $t = t_{in}$;
Output:
$A_{t_c}^{(M)}$ is a generator of the M-order approximation of the TEO;
$\psi(t_{out})$ is a wave packet at time $t = t_{out}$;
$\alpha_\zeta^{(M)}$, $\zeta = 1, ..., M$, are weights of the operator-difference scheme (15);
Local:
$eq_j,\ j = 1, ..., L$ are auxiliary equations ($L = 2M$);
$A_{(j)}(t),\ j = 1, ..., L$, is a generator of the j-order approximation of the TEO;
$\dot{A}_{(j)}(t),\ j = 1, ..., L$, are the partial derivatives of the $A_{(j)}(t)$ with respect to t;
$F^{(j)}(A_{(i)}(t), 1 \le i \le j-1; H(t))$, $j = 1, ..., L$, are auxiliary functional solutions;
$t_i,\ i = 0, ..., L$, are auxiliary time variables;
$t_c = t_{in} + \tau/2$ is a middle point of time intervals;
$\partial_t^j H(t_c) \equiv \partial_t^j H(x,t)\Big|_{t=t_c}$, $j = 0, ..., L$, are the partial derivatives of the order j of the $H(t)$ with respect to t at $t = (t_c)$;
$\psi^{\zeta/M}$ at $\zeta = 0, ..., M$ are auxiliary functions;

```
1: for $j$:=1 to $L$ do
        $eq_j := \sum_{n=1}^{j} \sum_{q=0}^{j-n} \sum_{l_1,\dots l_q \ge 1}^{n+\Sigma_{i=1}^q l_i = j} \frac{[A_{(l_1)}(t), [A_{(l_2)}(t), [..., [A_{(l_q)}(t), \dot{A}_{(n)}(t)]...]]]}{(q+1)!}$;
   end for;
   $eq_1 := eq_1 + \sqrt{-1}H(t)$;
2: $eq_j = 0,\ j = 1, ..., L$    $\rightarrow$   $\dot{A}_{(j)}(t) = F^{(j)}(A_{(i)}(t), 1 \le i \le j-1; H(t))$;
3: for $j$:=1 to $L$ do
        $H(t_{j-1}) := subs(t \rightarrow t_{j-1}, H(t))$;
        $A_{(j)}(t_j) := \int_{t_{in}}^{t_j} dt_{j-1} F^{(j)}(A_{(i)}(t_{j-1}), 1 \le i \le j-1; H(t_{j-1}));$
        if $L > 1$ then
          for $i := j+1$ to $L$ do
            $A_{(j)}(t_i) := subs(t_j \to t_i, A_{(j)}(t_j));$
          end for
        end if
  end for;
4: $A_{(L)}(t_L) := \sum_{j=1}^{L} A_{(j)}(t_L);$
5: $A_{(L)}(t_L) := subs(H(t_i) \to \sum_{j=0}^{L} \frac{(t_i - t_c)^j}{j!} \partial_t^j H(t_c), i = 1, ..., L-1, \frac{1}{\tau} A_{(L)}(t_L));$
6: $A_{t_c}^{(M)} := subs(t_L \to t_{out}, A_{(L)}(t_L));$
7: ${}_1F_1(-M, -2M, 2M\imath/\alpha) = 0 \quad \to \quad \alpha_\zeta^{(M)}, \quad \zeta = 1, 2, \ldots, M;$
8: $\psi^0 := \psi(t_{in});$
  for $\zeta$:=1 to $M$ do
      $\left(I + \frac{\tau}{2M}\overline{\alpha}_\zeta^{(M)} A_{t_c}^{(M)}\right)\psi^{\zeta/M} = \left(I + \frac{\tau}{2M}\alpha_\zeta^{(M)} A_{t_c}^{(M)}\right)\psi^{(\zeta-1)/M} \to \psi^{\zeta/M};$
  end for;
  $\psi(t_{out}) := \psi^1;$
```

Remarks:

1. In the pseudo-code expression of step **1:** the definition of eq_j comes from the identity (5), where Table 1 shows a range of the summation indexes (n, q, l_1, ..., l_q, j) in step **1:**.

2. Integrals appearing at step **3:** are evaluated explicitly at step **5:** after substitution of the Taylor-series expansion of the Hamiltonian $H(x,t)$ in a vicinity of the point $t = t_c = t_{in} + \tau/2$.

3 Unitary Schemes with a Partial Splitting of the Evolution Operator

Let us consider the Cauchy problem,

$$\imath \frac{\partial}{\partial t}\psi(x,\, t) = H(x,t)\psi(x,\, t), \quad (x,t) \in \mathbf{R} \times [t_0, T], \tag{16}$$
$$H(x,t) = H(x) + q(x,t), \quad H(x) = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + V(x),$$
$$\psi(-\infty,t) = \psi(\infty,t) = 0, \quad \psi(x,t_0) = \psi_0(x),$$

of the TDSE in the interval for an atomic model in an external field.

For the operators, $q(x,t)$ and $H(x)$, and the wave function $\psi(x)$, we make the following assumptions. The $q(x,t)$ describes the dipole-approximation of the interaction between the atom and the external field $f(t)$, which is written in the form, $q(x,t) = f(t)x$. The $H(x)$ is the Hamiltonian of the atom without the effect of the external field. The $H(x)$ admits the non-negative continuous spectra and/or non-positive discrete spectra: With a continuous spectral value $E_c \geq 0$ (resp. a discrete discrete spectral value E_d, eigenfunctions denoted by $\psi_{E_c}(x)$ (resp. $\psi_{E_d}(x)$) are associated. As the initial state, namely the Cauchy data, we choose the Gaussian packet $\psi_0(x) \in \mathbf{H^1}$ or a bound state $\psi_{E_d}(x)$. The normalization condition,

$$||\psi||^2 = \int_{-\infty}^{\infty} |\psi(x,t)|^2 dx = \int_{-\infty}^{\infty} \psi^*(x,t)\psi(x,t)dx = 1, \tag{17}$$

is claimed at any $t \geq t_0$.

Second-order scheme at the time step ($M = 1$): We wish to construct the second-order scheme in the time step τ for the Cauchy problem (16) on a uniform grid (2). For $M = 1$, the scheme (15) of the order $O(\tau^2)$ reads as well-known Cranck-Nicholson method for the Cauchy problem (16) with the effective operator,

$$\begin{aligned}
&\hat{A}_k^{(1)} \equiv A_k^{(1)} = H(x, t_{k+1/2}), \quad t_{k+1/2} = t_k + \frac{1}{2}\tau, \\
&(I + \frac{\tau}{2}\overline{\alpha}_1^{(1)}\hat{A}_k^{(1)})\psi^{k+1} = (I + \frac{\tau}{2}\alpha_1^{(1)}\hat{A}_k^{(1)})\psi^k, \\
&\psi^0 = \psi(x,t_0), \quad ||\psi^{k+1}|| = ||\psi^k|| \quad (k = 0, 1, \cdots, K-1).
\end{aligned} \tag{18}$$

In these expressions $\alpha_1^{(1)}$ is given in Table 2. As is shown in the third line of (18), the norm-preserving claim (17) is ensured in this scheme.

Fourth-order scheme at the time step ($M = 2$): In the case of $M = 2$, we apply the gauge transformation, $\psi^k \to \hat{\psi}^k = \exp\left(\imath\tau^2 S_k^{(2)}\right)\psi^k$, to the wave function, where $S_k^{(2)}$ is chosen to be

$$\imath\tau^2 S_k^{(2)} = \frac{\imath\tau^2}{12}\frac{\partial q(x, t_{k+1/2})}{\partial t}. \tag{19}$$

Then the gauge transformation above and Eq. (10) are put together to show that the new effective operator, $\hat{A}_k^{(2)} = \exp\left(\imath\tau^2 S_2\right) A_k^{(2)} \exp\left(-\imath\tau^2 S_2\right)$, induced by the gauge transformation takes the form,

$$\hat{A}_k^{(2)} = A_k^{(2)} + \frac{\imath\tau^2}{12}[\dot{H}, H] = H(x, t_{k+1/2}) + \frac{\tau^2}{24}\frac{\partial^2 q(x, t_{k+1/2})}{\partial t^2}. \tag{20}$$

As is seen above, the characteristic of our gauge transformation is that all the coefficients in the expansion of $\hat{A}_k^{(2)}$ up to order $O(\tau^4)$ become real-valued after the transformation. The algorithm combined with the gauge transformation starting from $\psi^0 = \psi(x, t_0)$ can read as follows:

$$\begin{array}{ll}
\mathbf{1}. & \hat{\psi}^k = \exp\left(\imath\tau^2 S_k^{(2)}\right)\psi^k, \\
\mathbf{2}. & (I + \frac{\tau}{4}\overline{\alpha}_1^{(2)}\hat{A}_k^{(2)})\hat{\psi}^{k+1/2} = (I + \frac{\tau}{4}\alpha_1^{(2)}\hat{A}_k^{(2)})\hat{\psi}^k, \\
\mathbf{3}. & (I + \frac{\tau}{4}\overline{\alpha}_2^{(2)}\hat{A}_k^{(2)})\hat{\psi}^{k+1} = (I + \frac{\tau}{4}\alpha_2^{(2)}\hat{A}_k^{(2)})\hat{\psi}^{k+1/2}, \\
\mathbf{4}. & \psi^{k+1} = \exp\left(-\imath\tau^2 S_k^{(2)}\right)\hat{\psi}^{k+1},
\end{array} \tag{21}$$

where $\alpha_1^{(2)}$ and $\alpha_2^{(2)}$ are calculated in Table 2. In this scheme, the norm-preserving claim (17) is ensured in the form, $||\hat{\psi}^{k+1}|| = ||\hat{\psi}^{k+1/2}|| = ||\hat{\psi}^k||$, $(k = 0, 1, \ldots, K-1)$.

Sixth-order scheme at the time step ($M = 3$): Like in the case of $M = 2$, we apply the gauge transformation, $\hat{\psi}^k = \exp\left(\imath(\tau^2 S_k^{(2)} + \tau^4 S_k^{(4)})\right)\psi^k$, where $S_k^{(4)}$ is chosen to be

$$\imath\tau^4 S_k^{(4)} = -\frac{\imath\tau^4}{720}\frac{\partial V(x)}{\partial x}\frac{\partial f(t_{k+1/2})}{\partial t} + \frac{\imath\tau^4}{480}x\frac{\partial^3 f(t_{k+1/2})}{\partial t^3}. \tag{22}$$

The gauge transformation above and Eq. (11) are put together to show that the new effective operator, $\hat{A}_k^{(3)}$, takes the form,

$$\begin{aligned}
\hat{A}_k^{(3)} &= \exp\left(\imath(\tau^2 S_k^{(2)} + \tau^4 S_k^{(4)})\right) A_k^{(3)} \exp\left(-\imath(\tau^2 S_k^{(2)} + \tau^4 S_k^{(4)})\right) \\
&= \hat{A}_k^{(2)} + \frac{\tau^4}{5760}\left[3x\frac{\partial^4 f(t_{k+1/2})}{\partial t^4} + 4f(t_{k+1/2})\frac{\partial^2 f(t_{k+1/2})}{\partial t^2}\right. \\
&\qquad \left. +4\left(\frac{\partial f(t_{k+1/2})}{\partial t}\right)^2 + 8\frac{\partial V(x)}{\partial x}\frac{\partial^2 f(t_{k+1/2})}{\partial t^2}\right],
\end{aligned}$$

so that all the coefficients in the expansion of $\hat{A}_k^{(2)}$ up to order $O(\tau^6)$ become real-valued after the transformation. The algorithm combined with the gauge transformation starting from $\psi^0 = \psi(x, t_0)$ can read as follows:

$$\begin{array}{ll}
\mathbf{1}. & \hat{\psi}^k = \exp\left(\imath(\tau^2 S_k^{(2)} + \tau^4 S_k^{(4)})\right)\psi^k, \\
\mathbf{2}. & (I + \frac{\tau}{6}\overline{\alpha}_1^{(3)}\hat{A}_k^{(3)})\hat{\psi}^{k+1/3} = (I + \frac{\tau}{6}\alpha_1^{(3)}\hat{A}_k^{(3)})\hat{\psi}^k, \\
\mathbf{3}. & (I + \frac{\tau}{6}\overline{\alpha}_2^{(3)}\hat{A}_k^{(3)})\hat{\psi}^{k+2/3} = (I + \frac{\tau}{6}\alpha_2^{(3)}\hat{A}_k^{(3)})\hat{\psi}^{k+1/3}, \\
\mathbf{4}. & (I + \frac{\tau}{6}\overline{\alpha}_3^{(3)}\hat{A}_k^{(3)})\hat{\psi}^{k+1} = (I + \frac{\tau}{6}\alpha_3^{(3)}\hat{A}_k^{(3)})\hat{\psi}^{k+2/3}, \\
\mathbf{5}. & \psi^{k+1} = \exp\left(-\imath(\tau^2 S_k^{(2)} + \tau^4 S_k^{(4)})\right)\hat{\psi}^{k+1},
\end{array} \tag{23}$$

where $\alpha_1^{(3)}$, $\alpha_2^{(3)}$ and $\alpha_3^{(3)}$ are listed in Table 2. In this scheme, the norm-preserving claim (17) is ensured in the form, $||\hat{\psi}^{k+1}|| = ||\hat{\psi}^{k+2/3}|| = ||\hat{\psi}^{k+1/3}|| = ||\hat{\psi}^k||$ $(k = 0, 1, \ldots, K-1)$.

4 Conversion to the Algebraic Problem by FEM

In the numerical calculations of the problem (16) on the grid (2) the boundary conditions in (16) with respect to the spatial variable are reduced to the finite interval $\psi(x_{min}, t) = \psi(x_{max}, t) = 0$. We consider a discrete representation of solutions $\psi(x;t)$ of problem (16) by means of FEM on the grid, $\Omega_h^p = (x_0 = x_{min}$, $x_j = x_{j-1} + h_j, x_{\bar{n}} = x_{max})$, in a finite sum in each $t = t_k$ of the grid $\Omega_\tau[t_0, T]$:

$$\psi(x;t) = \sum\nolimits_{\mu=0}^{\bar{n}p} \chi_\mu(t) N_\mu(x) = \sum\nolimits_{r=0}^{\bar{n}} \sum\nolimits_{j=1}^{p} \chi_{r+p(j-1)}(t) N_{r+p(j-1)}^p(x), \quad (24)$$

where $N_\mu(x)$ are local functions and $\chi_\mu(t)$ are node values of $\psi(x;t)$. The local functions $N_\mu(x)$ are piecewise polynomials of the given order p which equals one only in the node x_μ and equals zero in all other nodes $x_\nu \neq x_\mu$, i.e., $N_\nu(x_\mu) = \delta_{\nu\mu}$, $\mu, \nu = 0, 1, ..., \bar{n}p$. The coefficients $\chi_\nu(t_k)$ are formally connected with solution $\psi(x_{j,r}^p; t_k)$ in a node $x_\nu = x_{j,r}^p$, $r = 1, ..., p$, $j = 0, ..., \bar{n}$:

$$\chi_\nu(t_k) = \chi_{r+p(j-1)}(t_k) \approx \psi(x_{j,r}^p; t_k), \quad x_{j,r}^p = x_{j-1} + \frac{h_j}{p} r.$$

The theoretical estimate for the $\mathbf{H^0}$ norm between the exact and numerical solution has the order of $||\psi - \chi|| = O(h^{p+1})$, where $h = \max_{1<j<\bar{n}} h_j$ is maximum step of grid [5]. It has been shown that we have a possibility to construct schemes with very high order of accuracy comparable with the computer one [8]. Let us consider the reduction of differential equations (15) on the interval $\Delta : \; x_{min} < x < x_{max}$ with boundary conditions at points x_{min} and x_{max} at each step k of the grid (2) rewriting in the form

$$\left(\mathbf{A}_\zeta(x, t_{k+1/2})\right) \psi^{k+\zeta/M}(x) = \left(\mathbf{B}_\zeta(x, t_{k+1/2})\right) \psi^{k+(\zeta-1)/M}(x), \quad (25)$$

where $\mathbf{A}$ and $\mathbf{B}$ are differential operators. Substituting expansion (24) to (25) and integrating with respect to x by parts in the interval $\Delta = \cup_{j=1}^{\bar{n}} \Delta_j$, we arrive at a system of the linear algebraic equations

$$\left(\mathbf{a}_\zeta(t_{k+1/2})\right)_{\mu\nu} \chi_\nu^{k+\zeta/M} = \mathbf{c}_\mu, \quad \mathbf{c}_\mu = \left(\mathbf{b}_\zeta(t_{k+1/2})\right)_{\mu\nu} \chi_\nu^{k+(\zeta-1)/M}, \quad (26)$$

within the framework of the briefly described FEM. Using p-order Lagrange elements [5], we present below an algorithm **AXeqC** for construction of algebraic problem (26) by the FEM in the form of conventional pseudocode. Its MAPLE realization allows us to explicitly recalculate the indices μ, ν and to test corresponding modules in FORTRAN code.

For solution of algebraic problem (26) with respect to unknown vector $\chi^{k+\zeta/M} \equiv \{\chi_\mu^{k+\zeta/M}\}^T$ at large values of dimension $\bar{n}p$ of matrix $\mathbf{a}$ and a given vector $\mathbf{c}$ one can use the subroutine F07BRF (ZGBTRS) of NAG Fortran Library Routine Document [13]. The main point of view to pay attention to the schemes with the partial splitting TEO derived in the previous section consists in preserving a symmetric structure of the band matrices (26) needed for applying the conventional subroutines of the FEM [6].

Algorithm AXeqC

Input:
$\Delta = \cup_{j=1}^{\bar n}\Delta_j = [x_{\min}, x_{\max}]$, is interval of changing of space variable x;
$h_j = x_j - x_{j-1}$ is a grid step;
$\bar n$ is a number of subintervals $\Delta_j = [x_{j-1}, x_j]$;
p is an order of finite elements;
$\left(\mathbf{A}_\zeta(x, t_{k+1/2})\right)$, $\left(\mathbf{B}_\zeta(x, t_{k+1/2})\right)$ are differential operators in Eq. (25);
Output:
N_μ is the basis functions in (24);
$\left(\mathbf{a}_\zeta(t_{k+1/2})\right)_{\mu\nu}$, $\left(\mathbf{b}_\zeta(t_{k+1/2})\right)_{\mu\nu}$ are matrix elements in system of algebraic equations (26);
Local:
$x_{j,r}^p$ are nodes;
$\phi_{j,r}^p(x)$ are Lagrange elements;
$\mu, \nu = 0, 1, ..., \bar n p$;

```
1: for $j$:=1 to $\bar n$ do
       for $r$:=0 to $p$ do
         $x_{j,r}^p = x_{j-1} + \frac{h_j}{p} r$
       end for;
   end for;
2: $\phi_{j,r}^p(x) = \dfrac{\prod_{k\neq r}(x - x_{j,k}^p)}{\prod_{k\neq r}((x_{j,r}^p - x_{j,k}^p))}$
3: $N_0(x) := \{\phi_{1,0}^p(x), x \in \Delta_1; 0, x \notin \Delta_1\}$;
   for $j$:=1 to $\bar n$ do
       for $r$:=1 to $p-1$ do
         $N_{r+p(j-1)}(x) := \{\phi_{j,r}^p(x), x \in \Delta_j; 0, x \notin \Delta_j, \}$
       end for;
     $N_{jp}(x) := \{\phi_{j,p}^p(x), x \in \Delta_j; \phi_{j+1,0}^p(x), x \in \Delta_{j+1}; 0, x \notin \Delta_j \bigcup \Delta_{j+1}\}$;
   end for;
   $N_{\bar n p}(x) := \{\phi_{\bar n,p}^p(x), x \in \Delta_{\bar n}; 0, x \notin \Delta_{\bar n}\}$;
4: for $\mu, \nu$:=0 to $\bar n p$ do
       $\left(\mathbf{a}_\zeta(t_{k+1/2})\right)_{\mu\nu} := \int_\Delta N_\mu(x)\left(\mathbf{A}_\zeta(x, t_{k+1/2})\right) N_\nu(x) dx$;
       $\left(\mathbf{b}_\zeta(t_{k+1/2})\right)_{\mu\nu} := \int_\Delta N_\mu(x)\left(\mathbf{B}_\zeta(x, t_{k+1/2})\right) N_\nu(x) dx$;
   end for;
```

Remarks:

1. For equation (18) matrix elements of the operator, $\left(\mathbf{A}_1(\rho, t_{k+1/2})\right)$:

$$\left(I + \frac{\tau}{2}\bar\alpha_1^{(1)}\hat A_k^{(1)}\right) = \left(I + \frac{\tau}{2}\bar\alpha_1^{(1)}\Big(-\frac{1}{2}\frac{\partial^2}{\partial x^2} + V(x) + q(x, t_{k+1/2})\Big)\right)$$

between local functions N_μ and N_ν defined in the same interval Δ_j are calculated by formula [3]

$$\left(\mathbf{a}_1(t_{k+1/2})\right)_{q+p(j-1),r+p(j-1)} = \int\limits_{-1}^{+1} \phi^p_{j,q}\phi^p_{j,r}\frac{h_j}{2}d\eta$$
$$+\frac{\bar{\alpha}_1^{(1)}\tau}{2}\int\limits_{-1}^{+1}\left\{\frac{2}{h_j^2}(\phi^p_{j,q})'(\phi^p_{j,r})' + \left(V(x)+q(x,t_{k+1/2})\right)\phi^p_{j,q}\phi^p_{j,r}\right\}\frac{h_j}{2}d\eta.$$

If integrals are not calculated analytically then we perform numerical evaluation [5], for example, by means of the Gauss quadrature formulae of the order $p+1$.

2. For solution of algebraic problem (26) with respect to unknown vector $\chi^{k+\zeta/M} \equiv \{\chi_\mu^{k+\zeta/M}\}^T$ at large values of dimension $\bar{n}p$ of matrix $\mathbf{a}$ and a given vector $\mathbf{c}$ a subroutine F07BRF (ZGBTRS) of NAG Fortran Library Routine Document[13] is used. Note, that for schemes with partial splitting from previous section are sufficient to apply the conventional subroutines for a symmetric band matrix [6].

5 Numerical Experiments

Oscillator in an external periodical field. Now we consider the problem (16) with the potential function $V(x) = \omega^2x^2/2$ that supports only a pure discrete spectrum. This potential describes a harmonic oscillator with the angular frequency ω in an external field $f(t) = f_0\sin(\omega_0 t)$ with strength f_0 and angular frequency ω_0. We choose the initial state $\psi_0(x)$ at the time $t_0 = 0$ as a Gaussian wave packet $\psi_0(x) = \sqrt[4]{\omega/\pi}\exp(-\omega(x-x_0)^2/2 + \imath p_0(x-x_0))$. For the numerical example considered below the constants are taken to be $x_0 = 0, p_0 = 1, \omega = \omega_0 = 1, f = 0.25$. This problem is a very good test for numerical experiments because it has the known analytical solution $\psi(x,t)$ [2].

In these experiments we consider the finite element grid Ω_h^p with 1000 elements of order $p = 6$ in the interval $[x_{min}, x_{max}]$, where $x_{min} = 20$, $x_{max} = 20$, and $0 \le t \le 10$. We use the enclosed three time grids $\Omega_\tau[0,10]$ with the step τ taking the values $\tau = 0.01, 0.005, 0.0025$ and examine the behavior of the function

$$Er^2(t;i) = \int_{x_{min}}^{x_{max}} [\psi(x,t) - \psi_i(x,t)]^*[\psi(x,t) - \psi_i(x,t)]dx, \qquad (27)$$

where the index $i = 1,2,3$ labels the numerical solutions, obtained for a different values of the time step τ. Having these three values of $Er(t;i)$, we can calculate the Runge ratio

$$\alpha_M(t) = \ln\frac{|Er(t;1) - Er(t,2)|}{|Er(t;2) - Er(t,3)|}/\ln 2. \qquad (28)$$

[3] Transformation to new variable η in integral ($x \mapsto \eta : \Delta_j = [x_{j-1}, x_j] \mapsto [-1,1]$) is given by formula $\eta = (x_j - x_{j-1})^{-1}[2x - (x_j + x_{j-1})]; x = 2^{-1}[(x_j - x_{j-1})\eta + (x_j + x_{j-1})], \quad h_j = x_j - x_{j-1}$.

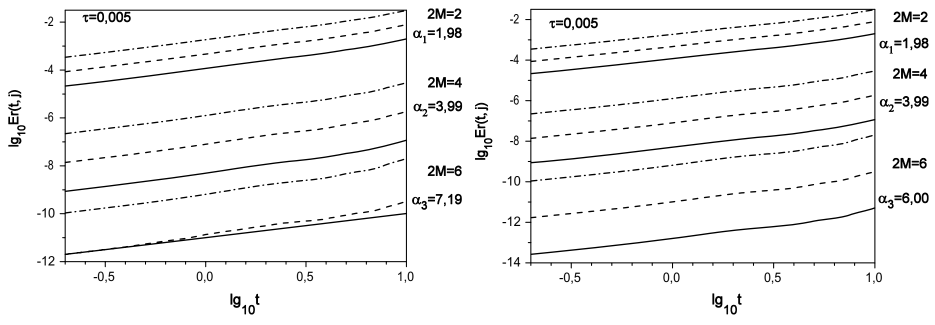


Fig. 1. Logarithm of discrepancy $\lg_{10} Er(t;i), i = 1,2,3$ (dash-dot, dashed and solid curves) for schemes with $M = 1,2,3$ calculated for oscillator in Fortran with double precision (15 significant digits) in left panel and quadruple one (33 significant digits) in right panel

Hence, we obtain the following theoretical estimates: $\alpha_1(t) \approx 2$ for the scheme (18), $\alpha_2(t) \approx 4$ for the scheme (21) and $\alpha_3(t) \approx 6$ for the scheme (23) Fig. 1 shows the plots of $Er(t;i), i = 1,2,3$ for these schemes (upper three curves correspond to the second-order scheme, middle three curves to the fourth-order scheme and lower three ones to the six-order scheme) and the mean value of α_M over all values $\alpha_M(t_k)$ of the grid $\Omega_\tau[0,10]$. One can see in Fig. 1 that for schemes starting from the order of $M \geq 3$ the floating-point calculations with at least a quadruple precision should be applied.

The Pöschl–Teller atom in a laser pulse field. Now we consider the propagation problem (16) that has oscillating solutions. In our point of view, to find directly such solutions with a given accuracy in an increasing interval of time with respect to a short duration time of a pulse field is a rather complicated problem[3]. To illustrate how the above approach allows an efficient solution of the TDSE problem, we consider a Pöschl–Teller atom (PTA) in a laser pulse electric field. For the PT model the potential function $V(x) = -\cosh^{-2} x$ supports only one bound state $\psi_0(x) = 1/\sqrt{2}\cosh x$, with the eigenvalue $E_0 = -0.5$ a.u., and a continuum of the known scattering states with $E > 0$. The laser pulse $f(t)$ is given by $f(t) = \left\{ f_0 \sin^2(\frac{\pi t}{2t_0}), 0 < t < 2t_0; 0, |t - t_0| \geq t_0, \right\}$ where $f_0 = t_0 = 1$. We choose the corresponding ground state $\psi_0(x)$ as an initial state. To approximate the solution $\psi_i(x,t)$, $i = 1,2,3,4$ we use 1600 finite elements with $p = 6$ and the finite element grid $\Omega = \{-1500(200) - 300(200) - 20(200) - 1(400)1(200)20(200)300(200)1500\}$, where the numbers in brackets denote the number of finite elements in the intervals. We calculated the above solution over the enclosed time grids $\Omega_\tau[t_0 = 0, T = 10]$ with four different time steps $\tau = 0.01, 0.005, 0.0025, 0.00125$. Fig. 2 displays the wave function calculated at time $T = 10$ and behavior of discrepancies $Er(t;i), i = 1,2,3$ evaluated by formulae (27) and (28) at $M = 1,2,3$ where the function $\psi_4(x,t)$ was used instead of an analytical solution $\psi(x,t)$. Here, we again obtain the numerical estimates of $\alpha_M(t)$ and their mean value, α_M, that strongly correspond to theoretical ones.

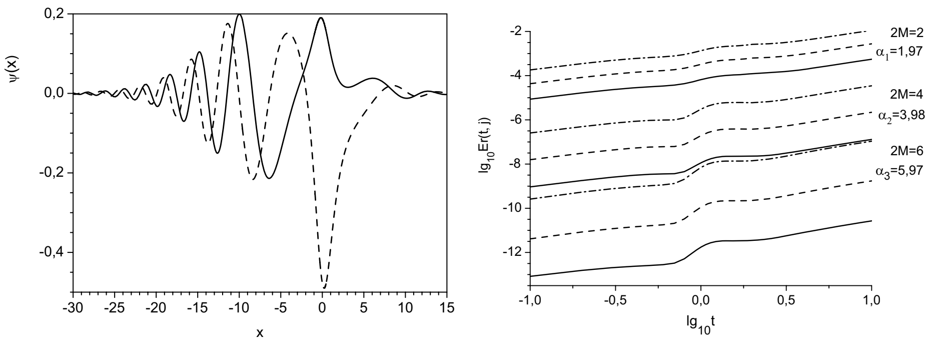


Fig. 2. Real and imaginary parts of solution $\phi(x,t)$ (solid and dashed curves) for PTA atom at $t = T = 10$ in left panel and logarithm of discrepancy $Er(t;i), i = 1,2,3$ (dash-dot, dashed and solid curves) for schemes with $M = 1,2,3$ calculated in Fortran by quadruple precision (33 significant digits) in right panel

6 Conclusions

We have presented a new computational approach to solve the TDSE, in which partial (unitary) splitting of evolution operator and the FEM are combined together effectively. Especially, to realize our approach in an explicit form, we have derived the second-, fourth-, and sixth-order approximations with respect to time. Several numerical results have been also given which turn out to agree with the theoretical ones to a good extent.

As our future program, we wish to mention an extension of our proposed approach to the nonlinear TDSE with the use of the Lie symmetry formalism [14,15], which some of the authors have a plausible reason to think of. If it is possible, 'Lie-admissible' TDSEs could be thought of, from which we could find exact solutions [16,17]. Our approach would be worth being applied to the quantum control problem, some pre-experimental calculations in the atomic dynamics in traps and/or external-pulse fields, and other quantum calculations [2].

The authors thanks O. Chuluunbaatar at JINR for discussions. This work was supported in part by Grant of RFBR Nos. 03-02-16263 and 04-01-00784, Grant of the President of the Bulgarian State Agency for Atomic Energy (2004), by Grant I-1402/2004 of the Bulgarian Foundation for Scientific Investigations, and by Grant-in-Aid Scientific Research Nos. 13660065 and 16560050 from JSPS.

References

1. Derbov, V.L., Melnikov, L.A., Umansky, I.M., Vinitsky, S.I.: Multipulse laser spectroscopy of p-bartHe$^+$: Measurement and control of the metastable state populations. Phys. Rev. A **55** (1997) 3394–3400
2. Butkovskiy, A.G., Samoilenko, Yu.I.: Control of Quantum-Mechanical Processes and Systems. Kluwer Academic Publishers, Dordrecht Hardbound (1990)

3. Bankov, N.G., Kaschiev, M.S., Vinitsky, S.I.: Adaptive method for solving the time-dependent Schrödinger equation. Comptes rendus de l'Academie bulgare des Sciences **55** (2002) 25–30
4. Derbov, V.L., Kaschiev, M.S.,. Serov, V.V, Gusev, A.A., Vinitsky, S.I.: Adaptive numerical methods for time-dependent Schroedinger equation in atomic and laser physics. Proc. SPIE **5069** (2003) 52–60
5. Strang, G., Fix, G.: An Analisys of The Finite Element Method. Englewood Cliffs, Prentice Hall, New York (1973)
6. Bathe, K.J.: Finite Element Procedures in Engineering Analysis. Englewood Cliffs, Prentice Hall, New York (1982)
7. Abrashkevich, A.G., Abrashkevich, D.G., Kaschiev, M.S., Puzynin, I.V.: Finite-element solution of the coupled-channel Schrödinger equation using high-order accuracy approximations. Comput. Phys. Commun. **85** (1995) 40–65
8. Abrashkevich, A.G., Kaschiev, M.S., Vinitsky, S.I.: A New Method for Soling an Eigenvalue Problem for a System Of Three Coulomb Particles within the Hyperspherical Adiabatic Representation. J. Comp. Phys. **163** (2000) 328–348
9. Puzynin, I.V., Selin A.V., Vinitsky, S.I.: A high-order accuracy method for numerical solving of the time-dependent Schrödinger equation. Comput. Phys. Commun. **123** (1999) 1–6
   Puzynin, I.V., Selin A.V., Vinitsky, S.I.: Magnus-factorized method for numerical solving the time-dependent Schrödinger equation. Comput. Phys. Commun. **126** (2000) 158–161
10. Magnus, W.: On the exponential solution of differential equations for a linear operator. Commun. Pure Appl. Math. **7** (1954) 649–673
    Wilcox, R.M.: Exponential operators and parameter differentiation in quantum physics. J. Math. Phys. **8** (1967) 962–982
11. Samarskiy, A.A.: Theory of difference schemes. Nauka, Moscow (1977) [in Russian]
12. Crank, J., Nicholson, P.: A practical method for numerical evaluation of solutions of partial differential equation of the heat conduction type. Proc Cambridge Philos. Soc. **43** (1947) 50–67
13. http://www.nag.co.uk/numeric/numerical_libraries.asp
14. Olver, P.J.: Application of Lie groups to differential equations. Springer-Verlag, New York, Tokyo (1986)
15. Steinberg, S.: Lie Series, Lie Transformations and Their Applications, Lie Methods in Optics, (J. Sa'chez Mondrago'n and K. B. Wolf, Eds.), Lecture Notes in Physics, Springer-Verlag, New York, Tokyo (1985) 45–103
16. Jannussis, A.: Quantum equations of motion in the finite-difference approach. Lett. al Nuovo Cim. **40** (1984) 250–256
    Jannussis, A.: Difference equations in the Lie-addmissible formulation. Lett. al Nuovo Cim. **42** (1985) 129–133
17. Ukolov, Yu.A., Chekanov, N.A., Gusev, A.A., Rostovtsev, V.A., Vinitsky, S.I., Uwano, Y.: LINA01: A REDUCE program for the normalization of polynomial Hamiltonians. Computer Physics Communications **166** (2005) 66–80

On Some Results of Investigation of Kirchhoff Equations in Case of a Rigid Body Motion in Fluid

Valentin Irtegov and Tatyana Titorenko

Institute of Systems Dynamics and Control Theory, SB RAS,
134, Lermontov str., Irkutsk, 664033, Russia
irteg@icc.ru

Abstract. Some results of analysis of Kirchhoff equations, which describe the motion of a rigid body in the ideal incompressible fluid, are presented. With respect to these equations, a problem is stated to obtain steady-state motions, invariant manifolds of steady-state motions (IMSMs), and to investigate their properties in the aspect of stability and stabilization of motion. Our methods of investigation are based on classical results obtained by Lyapunov [1]. The computer algebra systems (CAS) "Mathematica", "Maple", and a software [2] are used as the tools. Lyapunov's sufficient stability conditions are derived for some steady-state motions obtained. A problem of optimal stabilization with respect to the first approximation equations is solved for some cases of unstable motion. This paper represents a continuation of our research, the results of which have been reported during CASC'2004 in St. Petersburg [3].

1 Introduction

Almost 250 years ago, L. Euler obtained differential equations, which describe motions of a rigid body with a fixed point in a moving coordinate system. Later these differential equations were generalized in numerous works, and are presently used for the purpose of modelling various mechanical systems: a system of rigid bodies in the gravitation field and other force fields, motions of a rigid body in the ideal fluid (Kirchhoff's equations), and also in hydrodynamic models of convection, chaos, Kolmogorov's flow in a channel with solid walls, etc. [4,5]. These equations are now used even as a system most suitable for demonstration of some methods of algebraic geometry, representation theory (Lie algebras, algebra of loops, Kac–Moody algebras), topology, which are applied in analysis of so called quite integrable systems [6]. Euler-type equations normally assume many algebraic first integrals, which may be used for obtaining information on qualitative properties of solutions of such equations. This paper discusses one of the types of above differential equations and presents some results, which are bound up with both obtaining steady-state solutions and invariant manifolds of steady motions and further investigation of their stability and stabilization by methods of the theory of Lyapunov functions, which is conducted with the aid of computer algebra tools.

In particular, a problem of a rigid body motion in the ideal fluid, which is described in terms of Kirchhoff equations, is considered. To wit, analysis of a recently revealed case of "quite integrability" of such a problem, for which the system Hamiltonian writes

$$2H = s_1^2 + s_2^2 + 2s_3^2 + 2\alpha r_1 s_3 - \alpha^2 r_3^2 = 2h,$$

and the differential equations assume other 3 first integrals, is conducted in the above aspect.

In obtaining steady-state motions and invariant manifolds we rely upon the following proposition.

Proposition 1. *If the differential equations of motion*

$$\dot{x}_i = X_i(t, x_1, \dots, x_n) \qquad (i = 1, \dots, n) \tag{1}$$

assume the first integral $V(t, x_1, \dots, x_n) = c$, then the system of equations

$$\frac{\partial V}{\partial x_i} = \varphi_i(t, x_1, \dots, x_n) = 0 \qquad (i = 1, \dots, n) \tag{2}$$

defines invariant manifolds of the initial system of differential equations (1).

Proof. Since $V(t, x_1, \dots, x_n)$ is the first integral of the system of differential equations (1), the following identity holds:

$$\sum_{i=1}^{n} \frac{\partial V}{\partial x_i} X_i + \frac{\partial V}{\partial t} = 0.$$

Having differentiated this identity with respect to x_j $(j = 1, \dots, n)$, after explicit transformations we have the following n equalities:

$$\sum_{i=1}^{n} \frac{\partial}{\partial x_i}\left(\frac{\partial V}{\partial x_j}\right) X_i + \frac{\partial}{\partial t}\left(\frac{\partial V}{\partial x_j}\right) = -\sum_{i=1}^{n} \frac{\partial V}{\partial x_i}\frac{\partial X_i}{\partial x_j} \qquad (j = 1, \dots, n).$$

As obvious from the latter system of equations, when (2) holds, left-hand sides of equations, which represent the derivatives of $\partial V/\partial x_j$ due to the differential equations (1), vanish on the manifold defined by the equations $\varphi_i(t, x_1, \dots, x_n) = 0$ $(i = 1, \dots, n)$. The latter allows us to state that the system of equations (2) defines invariant manifolds of the system of differential equations (1).

If the functions $\varphi_i(t, x_1, \dots, x_n)$ of the system of equations (2) are independent, then equations (2), allow one, generally speaking, to find out some set of steady-state solutions: $x_{1,k} = x_{1,k}(t), \dots, x_{n,k} = x_{n,k}(t)$. And under the condition of degeneracy of the scrutinized system (when $\varphi_i(t, x_1, \dots, x_n)$ are dependent), in the capacity of its solutions we obtain the invariant manifolds which are henceforth called invariant manifolds of steady motions (IMSMs). □

To the end of obtaining sufficient conditions of stability of steady-state solutions and IMSMs, theorems of Lyapunov's second method, in particular, the Routh–Lyapunov's theorem [7], are used. For the purpose of solving problems of optimal stabilization, the well-known N.N.Krassovski's theorem [8] is employed.

2 Obtaining Steady-State Motions

Kirchhoff's equations for a new integrable case [9] on account of the linear transformation of the variables [10] writes:

$$\begin{aligned} \dot r_1 &= (\alpha r_1 + 2s_3)r_2 - r_3 s_2, & \dot s_1 &= (\alpha r_1 + s_3)s_2 - \alpha^2 r_2 r_3, \\ \dot r_2 &= r_3 s_1 - r_1(\alpha r_1 + 2s_3), & \dot s_2 &= (\alpha r_3 - s_1)(\alpha r_1 + s_3), \\ \dot r_3 &= r_1 s_2 - r_2 s_1, & \dot s_3 &= -\alpha r_2 s_3. \end{aligned} \tag{3}$$

Here $s = (s_1, s_2, s_3)$ $r = (r_1, r_2, r_3)$ are vectors of "impulsive moment" and of "impulsive force", respectively, α is an arbitrary constant.

The system (3) assumes the four algebraic first integrals:

$$\begin{aligned} &2H = (s_1^2 + s_2^2 + 2s_3^2) + 2\alpha r_1 s_3 - \alpha^2 r_3^2 = 2h, \\ &V_1 = s_1 r_1 + s_2 r_2 + s_3 r_3 = c_1, \\ &V_2 = r_1^2 + r_2^2 + r_3^2 = c_2, \\ &9V_3 = \alpha^2((\alpha r_1 s_1 + \alpha r_2 s_2 + s_1 s_3)^2 + s_3^2(s_2^2 + (\alpha r_1 + s_3)^2)) = 9c_3, \end{aligned} \tag{4}$$

where H represents the system Hamiltonian "body-fluid".

For the purpose of finding out steady-state solutions for the system (3) and investigation of their stability, we – likewise in the previous paper [3] – use the Routh–Lyapunov's method [11], what allows us to perform a substantial part of computations with the aid of computer algebra systems.

While following the Routh–Lyapunov's method, let us restrict our consideration to linear combinations of integrals (4). Specifically, let us compose a full linear bundle of the first integrals of the problem:

$$K = \lambda_0 H - \lambda_1 V_1 - \lambda_2 V_2 - \lambda_3 V_3 \tag{5}$$

and write down for it the steady-state conditions with respect to all the variables:

$$\begin{aligned} \frac{\partial K}{\partial s_1} &= \frac{9}{2}\lambda_0 s_1 - \frac{9}{2}\lambda_1 r_1 - \alpha^4\lambda_3 r_1^2 s_1 - \alpha^4\lambda_3 r_1 r_2 s_2 - 2\alpha^3\lambda_3 r_1 s_1 s_3 - \alpha^3\lambda_3 r_2 s_2 s_3 \\ &\quad -\alpha^2\lambda_3 s_1 s_3^2 = 0, \\ \frac{\partial K}{\partial s_2} &= \frac{9}{2}\lambda_0 s_2 - \frac{9}{2}\lambda_1 r_2 - \alpha^4\lambda_3 r_1 r_2 s_1 - \alpha^4\lambda_3 r_2^2 s_2 - \alpha^3\lambda_3 r_2 s_1 s_3 - \alpha^2\lambda_3 s_2 s_3^2 = 0, \\ \frac{\partial K}{\partial s_3} &= \frac{9}{2}\alpha\lambda_0 r_1 - \frac{9}{2}\lambda_1 r_3 - \alpha^3\lambda_3 r_1 s_1^2 - \alpha^3\lambda_3 r_2 s_1 s_2 + 9\lambda_0 s_3 - \alpha^4\lambda_3 r_1^2 s_3 \\ &\quad -\alpha^2\lambda_3 s_1^2 s_3 - \alpha^2\lambda_3 s_2^2 s_3 - 3\alpha^3\lambda_3 r_1 s_3^2 - 2\alpha^2\lambda_3 s_3^3 = 0, \\ \frac{\partial K}{\partial r_1} &= \frac{9}{2}\alpha\lambda_0 s_3 - 9\lambda_2 r_1 - \frac{9}{2}\lambda_1 s_1 - \alpha^4\lambda_3 r_1 s_1^2 - \alpha^4\lambda_3 r_2 s_1 s_2 - \alpha^3\lambda_3 s_1^2 s_3 \\ &\quad -\alpha^4\lambda_3 r_1 s_3^2 - \alpha^3\lambda_3 s_3^3 = 0, \\ \frac{\partial K}{\partial r_2} &= 9\lambda_2 r_2 + \frac{9}{2}\lambda_1 s_2 + \alpha^4\lambda_3 r_1 s_1 s_2 + \alpha^4\lambda_3 r_2 s_2^2 + \alpha^3\lambda_3 s_1 s_2 s_3 = 0, \\ \frac{\partial K}{\partial r_3} &= (\alpha^2\lambda_0 + 2\lambda_2)r_3 + \lambda_1 s_3 = 0. \end{aligned} \tag{6}$$

In accordance with the **Proposition 1** given in the Introduction, solutions of system (6) define IMSMs of system (3) (in particular, steady-state motions). In the general case, these steady-state solutions may contain parameters λ_i, which enter in the family of integrals K (5), and hence may represent a family of steady-state solutions. Therefore, for the purpose of solving the stated problem (obtaining IMSMs for the system (3) corresponding to the family of first integrals K) it is necessary to solve a system of 6 nonlinear algebraic equations containing four parameters. The maximum power of the equations entering the system is 3.

To the end of obtaining solutions of system (6), let us employ sufficiently efficient methods of solving such problems – the Gröbner bases method and the method of elimination of variables with the use of the resultants. Let us preliminarily transform our equations by reducing their number.

In this connection, note that the 6th equation of system (6) is linear. The 4th and the 5th equations are also linear with respect to r_1 and r_2. Let us now use the above equations to remove r_1, r_2, and r_3 from the rest of equations (6). As a result, the steady-state equations write:

$$f_1(s_1, s_2, s_3) = 0;\ \ s_2 f_2(s_1, s_2, s_3) = 0;\ \ f_3(s_1, s_2, s_3) = 0. \tag{7}$$

Here we denote by f_1, f_2, f_3 the polynomials of the variables s_1, s_2, s_3. In the present paper we will not need explicit form of these equations.

As a result, the problem is reduced to investigation of three nonlinear algebraic equations dependent on the variables s_1, s_2, s_3 and parameters $\lambda_0, \lambda_1, \lambda_2, \lambda_3$, and α. The powers of the equations are, respectively, 9, 10 and 11. Consider the case when $s_2 = 0$. In this case, equations (7) are reduced to the following two ones, and after introduction of the denotation $a = 9\lambda_2,\ b = 2a + 9\alpha^2\lambda_0$ they will write:

$$\begin{aligned}
&a(2a\lambda_0 + 9\lambda_1^2)s_1 + 4a\alpha^4\lambda_0\lambda_3 s_1^3 + 2\alpha^8\lambda_0\lambda_3^2 s_1^5 - 9a\alpha\lambda_0\lambda_1 s_3 + 3\alpha^3(2a + 3\alpha^2\lambda_0)\\
&\times\lambda_1\lambda_3 s_1^2 s_3 + 2\alpha^7\lambda_1\lambda_3^2 s_1^4 s_3 + \alpha^2(9\alpha^2(\lambda_1^2 - \alpha^2\lambda_0^2) - \frac{4}{9}a^2)\lambda_3 s_1 s_3^2 + 4\alpha^8\lambda_0\lambda_3^2 s_1^3 s_3^2\\
&+\alpha^3(2a - 9\alpha^2\lambda_0)\lambda_1\lambda_3 s_3^3 + 4\alpha^7\lambda_1\lambda_3^2 s_1^2 s_3^3 + 2\alpha^8\lambda_0\lambda_3^2 s_1 s_3^4 + 2\alpha^7\lambda_1\lambda_3^2 s_3^5 = 0,
\end{aligned}$$

$$\begin{aligned}
&9ab\alpha\lambda_0\lambda_1 s_1 + \alpha^3(81\alpha^4\lambda_0^2 - 4a^2)\lambda_1\lambda_3 s_1^3 - 2b\alpha^7\lambda_1\lambda_3^2 s_1^5 - a(8a^2\lambda_0 + 81\alpha^4\lambda_0^3 + 2a\\
&\times(3\alpha^2\lambda_0^2 + \lambda_1^2))s_3 - \alpha^2(4a^2\alpha^2\lambda_0 - \frac{8}{9}a^3 + 81\alpha^4\lambda_0(\alpha^2\lambda_0^2 - \lambda_1^2) + 18a\alpha^2(3\alpha^2\lambda_0^2\\
&+\lambda_1^2))\lambda_3 s_1^2 s_3 + 2\alpha^6(\frac{2}{9}ab - 9\alpha^2\lambda_1^2)\lambda_3^2 s_1^4 s_3 - 3\alpha^3 b(2a + 3\alpha^2\lambda_0)\lambda_1\lambda_3 s_1 s_3^2 - 4\alpha^7 b\lambda_1\\
&\times\lambda_3^2 s_1^3 s_3^2 + 4a\alpha^2(\frac{2}{9}ab - 9\alpha^2\lambda_1^2)\lambda_3 s_3^3 + 4\alpha^6(\frac{2}{9}ab - 81\alpha^2\lambda_1^2)\lambda_3^2 s_1^2 s_3^3 - 2\alpha^7 b\lambda_1\lambda_3^2 s_1 s_3^4\\
&+2\alpha^6(\frac{2}{9}ab - 9\alpha^2\lambda_1^2)\lambda_3^2 s_3^5 = 0.
\end{aligned} \tag{8}$$

The power of both above equations is 5.

Now let us compute the resultant of (8) with respect to the variable s_1. After the factorization, it writes:

$$Res = (\alpha^2\lambda_0^2 + \lambda_1^2 + 2\lambda_0\lambda_2)s_3(\alpha^2(\alpha^4\lambda_0^2 + 4\lambda_2^2 + \alpha^2(\lambda_1^2 + 4\lambda_0\lambda_2)\lambda_3 s_3^2 + 9\lambda_1^2\lambda_2))^4$$
$$(a_0 s_3^8 + a_1 s_3^6 + a_2 s_3^4 + a_3 s_3^2 + a_4), \qquad (9)$$

where a_0, a_1, a_2, a_3, a_4 are polynomials of the parameters $\lambda_0, \lambda_1, \lambda_2, \lambda_3$. These are too bulky, and so are omitted herein.

When using (9), it is possible to estimate the number of roots for the system (8).

From the structure of the expression for *Res* and from the values of the respective coefficients, it is obvious that, in the general case, the degree of this polynomial with respect to s_3 is 17. Consequently, the resultant equated to zero allows us to obtain 17 values for s_3. If all of the values are real then, having substituted any of them into one of equations (8), we obtain a 5th-degree equation with respect to s_1. If the latter equation has 5 real solutions with respect to s_1 for each of 17 roots s_3, then the system (8) has 85 real solutions. Since, as a rule, our solutions contain the parameters, there arises a problem of (i) bifurcations of such families of solutions and (ii) analysis of their subfamilies under various restrictions imposed on the system parameters. Consider the procedure of obtaining some of these solutions.

Having equated the resultant to zero, consider now the following 3 possibilities.

Let us start from the case when $s_3 = 0$. Substitution of this value of s_3 into (8) entails in the following two equations containing s_1 :

$$\begin{aligned} &\mathrm{s}_1(81\lambda_1^2\lambda_2 + 162\lambda_0\lambda_2^2 + 36\alpha^4\lambda_0\lambda_2\lambda_3 s_1^2 + 2\alpha^8\lambda_0\lambda_3^2 s_1^4) = 0,\\ &\mathrm{s}_1(2\alpha^2\lambda_3 s_1^2 - 9\lambda_0)(9\lambda_2 + \alpha^4\lambda_3 s_1^2) = 0. \end{aligned} \qquad (10)$$

Under arbitrary values of λ_i the equations have only the zero common root $s_1 = 0$. To the end of finding other solutions, let us construct the Gröbner basis for system (10) with respect to both the variable s_1 and the parameter λ_2:

$$\begin{aligned} &\lambda_2 s_1(\alpha^4\lambda_0^3 + 2(2\alpha^2\lambda_0^2 + \lambda_1^2)\lambda_2 + 4\lambda_0\lambda_2^2) = 0,\\ &s_1(9(2\alpha^2\lambda_0^2 + \lambda_1^2)\lambda_2 + 18\lambda_0\lambda_2^2 + \alpha^6\lambda_0^2\lambda_3 s_1^2) = 0. \end{aligned} \qquad (11)$$

As obvious from direct substitution, equations (11) have the following solutions with respect to s_1, λ_2:

$$\Big\{\{s_1 = 0\},\ \{s_1 = -\frac{3\sqrt{\lambda_0}}{\sqrt{2}\alpha\sqrt{\lambda_3}},\ \lambda_2 = -\frac{2\alpha^2\lambda_0^2 + \lambda_1^2 - \lambda_1\sqrt{4\alpha^2\lambda_0^2 + \lambda_1^2}}{4\lambda_0}\},$$
$$\{s_1 = -\frac{3\sqrt{\lambda_0}}{\sqrt{2}\alpha\sqrt{\lambda_3}},\ \lambda_2 = -\frac{2\alpha^2\lambda_0^2 + \lambda_1(\lambda_1 + \sqrt{4\alpha^2\lambda_0^2 + \lambda_1^2})}{4\lambda_0}\},$$
$$\{s_1 = \frac{3\sqrt{\lambda_0}}{\sqrt{2}\alpha\sqrt{\lambda_3}}, \lambda_2 = -\frac{2\alpha^2\lambda_0^2 + \lambda_1^2 - \lambda_1\sqrt{4\alpha^2\lambda_0^2 + \lambda_1^2}}{4\lambda_0}\},$$
$$\{s_1 = \frac{3\sqrt{\lambda_0}}{\sqrt{2}\alpha\sqrt{\lambda_3}},\ \lambda_2 = -\frac{2\alpha^2\lambda_0^2 + \lambda_1(\lambda_1 + \sqrt{4\alpha^2\lambda_0^2 + \lambda_1^2})}{4\lambda_0}\}\Big\}.$$

Now, acting similarly, let us obtain solutions for equations (8), which correspond to the roots of the second multiplier in the expression for the resultant: $(\alpha^2(\alpha^4\lambda_0^2 + 4\lambda_2^2 + \alpha^2(\lambda_1^2 + 4\lambda_0\lambda_2)\lambda_3 s_3^2 + 9\lambda_1^2\lambda_2))^4$.

As regards s_3, the latter polynomial has two roots of multiplicity 4 each:

$$\Big\{\{s_3 = \frac{3\lambda_1\sqrt{\lambda_2}}{\sqrt{a}}\},\ \{s_3 = -\frac{3\lambda_1\sqrt{\lambda_2}}{\sqrt{a}}\}\Big\}.$$

Here $a = -\alpha^2(\alpha^4\lambda_0^2 + 4\lambda_2^2 + \alpha^2(\lambda_1^2 + 4\lambda_0\lambda_2))\lambda_3$.

Now substitute the first of the roots into (8), and for the equations obtained as a result of such a substitution construct the Gröbner basis with respect to both the variable s_1 and the parameter λ_1. This allows, under definite restrictions imposed on the parameters λ_i, to obtain up to 12 values of s_1 (including multiple ones) for the indicated value of s_3. There were five values of s_1 found out for the second root.

Finally, for the purpose of investigation of the 8th degree bi-polynomial (for the fourth multiplier in the expression of the resultant), standard techniques of solving the 4th degree algebraic equations were used. In this case, the results of computations turned out practically rather bulky, and the problem remained incomplete, open.

As a result (considering the multiplicity) over 70 families of solutions for the system (8) were found out. For obvious reasons, only real solutions are of interest for us. On the whole, 19 various real solutions were obtained.

Having the solutions for s_1, s_2, s_3, the respective values for the variables r_1, r_2, r_3 may be found from equations (6). Some of the solutions for the system (6) obtained by above technique are given below.

$$\begin{aligned}
&\Big\{\{s_1 = -\frac{6\lambda_2^{3/2}}{\alpha}, s_2 = 0, s_3 = 0, r_1 = 0, r_2 = 0, r_3 = 0, \lambda_0 = 0, \lambda_1 = 0,\\
&\lambda_3 = -\frac{1}{4\alpha^2\lambda_2^2}\}, \{s_1 = \frac{3\sqrt{\lambda_2}(\alpha^2\lambda_0 + 2\lambda_2)}{\alpha}, s_2 = 0, s_3 = \pm 3\sqrt{-\lambda_0\lambda_2(\alpha^2\lambda_0 + 2\lambda_2)},\\
&r_1 = \mp\frac{3\sqrt{-\lambda_0\lambda_2(\alpha^2\lambda_0 + 2\lambda_2)}}{\alpha}, r_2 = 0, r_3 = 3\lambda_0\sqrt{\lambda_2}, \lambda_1 = \pm\sqrt{-\lambda_0(\alpha^2\lambda_0 + 2\lambda_2)},\\
&\lambda_3 = -\frac{1}{2\alpha^4\lambda_0\lambda_2 - 4\alpha^2\lambda_2^2}\},\ \{s_1 = \mp\frac{3\sqrt{\lambda_0}}{\sqrt{2}\alpha\sqrt{\lambda_3}}, s_2 = 0, s_3 = 0, \qquad (12)\\
&r_1 = \pm\frac{3(\lambda_1^2 + \sqrt{4\alpha^2\lambda_0^2\lambda_1^2 + \lambda_1^4})}{2\sqrt{2}\alpha^3\lambda_1\sqrt{\lambda_0\lambda_3}}, r_2 = 0, r_3 = 0, \lambda_2 = -\frac{1}{4\lambda_0}(2\alpha^2\lambda_0^2 + \lambda_1^2\\
&-\sqrt{4\alpha^2\lambda_0^2\lambda_1^2 + \lambda_1^4})\},\ \{s_1 = \mp\frac{3\sqrt{\lambda_0}}{\sqrt{2}\alpha\sqrt{\lambda_3}}, s_2 = 0, s_3 = 0, r_1 = \pm\frac{3}{2\sqrt{2}\alpha^3\lambda_1\sqrt{\lambda_0\lambda_3}}(\lambda_1^2\\
&-\sqrt{4\alpha^2\lambda_0^2\lambda_1^2 + \lambda_1^4}), r_2 = 0, r_3 = 0, \lambda_2 = -\frac{2\alpha^2\lambda_0^2 + \lambda_1^2 + \sqrt{4\alpha^2\lambda_0^2\lambda_1^2 + \lambda_1^4}}{4\lambda_0}\}\Big\}.
\end{aligned}$$

3 Obtaining Invariant Manifolds of Steady-State Motions

Besides the problem of finding steady-state solutions for the system (3), a problem of obtaining IMSMs for the same equations was considered. In accordance with [12], equality of the Jacobi matrix determinant of system (6) to zero is the condition of existence of such IMSMs. In the general case, such an investigation entails in quite bulky expressions. Let us discuss a simpler problem. Instead of the Jacobian of system (6), consider the Jacobian of system (8), which for $\lambda_2 = -(\alpha^2\lambda_0^2 + \lambda_1^2)/(2\lambda_0)$ (9) assumes a rather compact form:

$$\begin{aligned} J = (\alpha\lambda_0 s_1 + \lambda_1 s_3)^2\{&81\lambda_0\lambda_1^4 - 432\alpha^5\lambda_0^3\lambda_1\lambda_3 s_1 s_3 - 324\alpha^3\lambda_0\lambda_1^3\lambda_3 s_1 s_3 + 96\alpha^7\lambda_0^2 \\ &\times\lambda_1\lambda_3^2 s_1 s_3(s_1^2 + s_3^2) + 4\alpha^8\lambda_0^3\lambda_3^2(5s_1^2 - 7s_3^2)(s_1^2 + s_3^2) \\ &+27\alpha^4(3\lambda_0^5 - 2\lambda_0^2\lambda_1^2\lambda_3 s_3^2) + 54\alpha^2(3\lambda_0^3\lambda_1^2 + \lambda_1^4\lambda_3(s_1^2 - 2s_3^2)) \\ &-4\alpha^6\lambda_0\lambda_3(27\lambda_0^3(s_1^2 - s_3^2) + \lambda_1^2\lambda_3(7s_1^2 - 5s_3^2)(s_1^2 + s_3^2))\}. \end{aligned} \tag{13}$$

It is obvious from (13) that this Jacobian vanishes, for example, when $s_1 = -\lambda_1 s_3/(\alpha\lambda_0)$. Having substituted the expression obtained for s_1 into (6), it is possible to find the respective values of the variables r_1, r_2, and r_3. The complete form of the solution writes:

$$\{r_1 = -\frac{s_3}{\alpha}, r_2 = 0, r_3 = \frac{\lambda_0 s_3}{\lambda_1}, s_1 = -\frac{\lambda_1 s_3}{\alpha\lambda_0}, s_2 = 0\}. \tag{14}$$

The vector field on the elements of the family of IMSMs (14) is given by the differential equation

$$\dot{s}_3 = 0$$

derived from equations (3) after removing s_1, s_2, r_1, r_2, r_3 from them with the aid of expressions (14). Therefore, the elements of the family of IMSMs obtained represent hyper-surfaces. The permanent helical motions of a rigid body correspond to the points of these hypersurfaces.

4 Stability Investigation of Steady-State Motions

The steady-state motions and the family of IMSMs (14) identified may be subject to stability investigation. The method of Lyapunov functions [1] is one of the traditional approaches in such cases, in particular, the above mentioned Routh–Lyapunov's method.

The procedure of obtaining sufficient conditions of stability of steady-state solutions by this technique practically reduces to the verification (in the simplest case) of sign-definiteness of the second variation of the integral K (5) in the neighborhood of the steady-state solution, which is of interest for us, on the manifold defined by the first variations each of $k-1$ integrals (of k integrals entering into the bundle K) equated to zero.

Consider the family of IMSMs (14) obtained above.

The second variation of K in the neighborhood of some helical motion, which lies on the chosen IMSM, represented in terms of deviations

$$z_1 = r_1 + \frac{s_3}{\alpha}, z_2 = r_2, z_3 = r_3 - \frac{\lambda_0 s_3}{\lambda_1}, z_4 = s_1 + \frac{\lambda_1 s_3}{\alpha\lambda_0}, z_5 = s_2, z_6 = s_3 - s_3^0$$

writes:

$$\delta^2 K = \frac{(\alpha^2\lambda_0^2+\lambda_1^2)(9\lambda_0-2\alpha^2\lambda_3 {s_3^0}^2)}{18\lambda_0^2} z_1^2 + \frac{\alpha^2\lambda_0^2+\lambda_1^2}{2\lambda_0} z_2^2 + \frac{\lambda_1^2}{2\lambda_0} z_3^2 - \lambda_1 z_1 z_4$$
$$+\frac{\lambda_0}{2} z_4^2 - \lambda_1 z_2 z_5 + (\frac{\lambda_0}{2} - \frac{\alpha^2\lambda_3 {s_3^0}^2}{9}) z_5^2,$$

and the respective variations of the first integrals H, V_1, V_2 can be written as:

$$\delta H = (\alpha z_1 - \frac{\alpha^2\lambda_0}{\lambda_1} z_3 - \frac{\lambda_1}{\alpha\lambda_0} z_4 + \frac{\lambda_1^4-\alpha^4\lambda_0^4}{\alpha^2\lambda_0^2\lambda_1^2} z_6) s_3^0 = 0,$$
$$\delta V_1 = (-\frac{\lambda_1}{\alpha\lambda_0} z_1 + z_3 - \frac{1}{\alpha} z_4 + 2(\frac{\lambda_0}{\lambda_1} + \frac{\lambda_1}{\alpha^2\lambda_0}) z_6) s_3^0 = 0,$$
$$\delta V_2 = 2(-\frac{1}{\alpha} z_1 + \frac{\lambda_0}{\lambda_1} z_3 + (\frac{1}{\alpha^2} + \frac{\lambda_0^2}{\lambda_1^2}) z_6) s_3^0 = 0.$$

Assuming that $s_3^0 \neq 0$, let us eliminate the variables z_1, z_4 with the use of the latter equations (because there are only two ones that are linearly independent) from $\delta^2 K$. As a result, the following quadratic form is obtained:

$$\delta^2\tilde{K} = \frac{\alpha^2\lambda_0^2+\lambda_1^2}{2\lambda_0} z_2^2 + \frac{9(\alpha^4\lambda_0^4+\alpha^2\lambda_0^2\lambda_1^2+\lambda_1^4) - 2\alpha^4\lambda_0(\alpha^2\lambda_0^2+\lambda_1^2)\lambda_3 {s_3^0}^2}{18\lambda_0\lambda_1^2} z_3^2$$
$$-\lambda_1 z_2 z_5 + (\frac{\lambda_0}{2} - \frac{\alpha^2\lambda_3 {s_3^0}^2}{9}) z_5^2 + \frac{\alpha^2}{9\lambda_0\lambda_1^3}(\alpha^2\lambda_0^2+\lambda_1^2)(9\lambda_0^3 - 2(\alpha^2\lambda_0^2$$
$$+\lambda_1^2)\lambda_3 {s_3^0}^2) z_3 z_6 + \frac{(\alpha^2\lambda_0^2+\lambda_1^2)^2(9\lambda_0^3 - 2(\alpha^2\lambda_0^2+\lambda_1^2)\lambda_3 {s_3^0}^2)}{18\lambda_0^2\lambda_1^4} z_6^2.$$

The conditions of sign-definiteness of $\delta^2\tilde{K}$ are known to be sufficient stability conditions for the helical motions, which belong to our IMSMs. When representing them in the form of Sylvester conditions we have:

$$\begin{aligned}
&1.\ \frac{\alpha^2\lambda_0^2+\lambda_1^2}{2\lambda_0} > 0,\\
&2.\ \frac{9(\alpha^6\lambda_0^6+2\alpha^4\lambda_0^4\lambda_1^2+2\alpha^2\lambda_0^2\lambda_1^4+\lambda_1^6) - 2\alpha^4\lambda_0(\alpha^2\lambda_0^2+\lambda_1^2)^2\lambda_3 {s_3^0}^2}{\lambda_0^2\lambda_1^2} > 0,\\
&3.\ \frac{\alpha^2}{\lambda_0^2\lambda_1^2}(2(\alpha^2\lambda_0^2+\lambda_1^2)\lambda_3 {s_3^0}^2 - 9\lambda_0^3)(2\alpha^4\lambda_0(\alpha^2\lambda_0^2+\lambda_1^2)\lambda_3 {s_3^0}^2 - 9(\alpha^4\lambda_0^4\\
&\quad +\alpha^2\lambda_0^2\lambda_1^2+\lambda_1^4)) > 0,\\
&4.\ \frac{(\alpha^2\lambda_0^2+\lambda_1^2)^3(9\lambda_0^3 - 2(\alpha^2\lambda_0^2+\lambda_1^2)\lambda_3 {s_3^0}^2)^2}{\lambda_0^2\lambda_1^2} > 0.
\end{aligned} \qquad (15)$$

A standard software package “Algebra‘ InequalitySolve‘” of CAS “Mathematica” is used for the purpose of verification of compatibility for this system of inequalities. Its application to (15) gives evidence that the inequalities are compatible when:

$$\begin{aligned}
&\alpha < 0 \vee \lambda_0 > 0 \\
&\vee\Big(\lambda_1 < 0 \vee \left(\lambda_3 \le 0 \wedge -\frac{3\lambda_0^{3/2}}{\sqrt{2}\sqrt{\alpha^2\lambda_0^2+\lambda_1^2}\sqrt{\lambda_3}} < s_3^0 < \frac{3\lambda_0^{3/2}}{\sqrt{2}\sqrt{\alpha^2\lambda_0^2+\lambda_1^2}\sqrt{\lambda_3}}\right) \\
&\wedge\lambda_1 > 0 \vee \left(\lambda_3 \le 0 \wedge -\frac{3\lambda_0^{3/2}}{\sqrt{2}\sqrt{\alpha^2\lambda_0^2+\lambda_1^2}\sqrt{\lambda_3}} < s_3^0 < \frac{3\lambda_0^{3/2}}{\sqrt{2}\sqrt{\alpha^2\lambda_0^2+\lambda_1^2}\sqrt{\lambda_3}}\right)\Big) \\
&\wedge\alpha > 0 \vee \lambda_0 > 0 \\
&\vee\Big(\lambda_1 < 0 \vee \left(\lambda_3 \le 0 \wedge -\frac{3\lambda_0^{3/2}}{\sqrt{2}\sqrt{\alpha^2\lambda_0^2+\lambda_1^2}\sqrt{\lambda_3}} < s_3^0 < \frac{3\lambda_0^{3/2}}{\sqrt{2}\sqrt{\alpha^2\lambda_0^2+\lambda_1^2}\sqrt{\lambda_3}}\right) \\
&\wedge\lambda_1 > 0 \vee \left(\lambda_3 \le 0 \wedge -\frac{3\lambda_0^{3/2}}{\sqrt{2}\sqrt{\alpha^2\lambda_0^2+\lambda_1^2}\sqrt{\lambda_3}} < s_3^0 < \frac{3\lambda_0^{3/2}}{\sqrt{2}\sqrt{\alpha^2\lambda_0^2+\lambda_1^2}\sqrt{\lambda_3}}\right)\Big).
\end{aligned}$$

Therefore, as far as elements of the family of IMSMs (14) are concerned, stable in the sense of Lyapunov are only those helical motions of the body for which parameter s_3^0 satisfies the latter conditions.

With the use of the Routh–Lyapunov's technique an attempt was made to obtain stability conditions for practically all the obtained solutions of equations (6). For most of these solutions the obtained sufficient stability conditions (Sylverster conditions) turned out to be incompatible for any values of parameters λ_i. In this connection, the problem of stabilization of such motions is of interest for us.

5 Optimal Stabilization of Steady-State Motions

In order to stay within the framework of Lyapunov's second method, a well-known N.N.Krassovski's theorem [8] is applied for solving problems of optimal stabilization.

Let there be the need to stabilize the equilibrium position $x_1 = 0, \dots, x_n = 0$ of the system of equations:

$$\dot{x}_i = X_i(t, x_1, \dots, x_n, u_1, \dots, u_r) \quad (i = 1, \dots, n) \tag{16}$$

for a given quality criterion:

$$J = \int_{t_0}^{\infty} \omega(t, x_1(t), \dots, x_n(t), u_1(t), \dots, u_r(t))dt. \tag{17}$$

Let us choose a Lyapunov function $V(t, x_1, \dots, x_n)$ and make up an expression:

$$\begin{aligned}
B[V, t, x_1, \dots, x_n, u_1, \dots, u_r] &= \frac{\partial V}{\partial t} + \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} X_i(t, x_1, \dots, x_n, u_1, \dots, u_r) \\
&+\omega(t, x_1, \dots, x_n, u_1, \dots, u_r).
\end{aligned} \tag{18}$$

Hence the following theorem holds.

Theorem 1. *If for the differential equations (16) it is possible to obtain a positive definite function $V^0(t, x_1, \dots, x_n)$ assuming an infinitesimal limit as well as the controls $u_j^0(t, x_1, \dots, x_n)$ $(j = 1, \dots, r)$, which satisfy the following conditions in the domain $t \geq 0$, $|x_i| \leq H$:*
1) the function

$$\omega(t, x_1, \dots, x_n) = \omega(t, x_1, \dots, x_n, u_1^0(t, x_1, \dots, x_n), \dots, u_r^0(t, x_1, \dots, x_n)) \quad (19)$$

is positive definite,
2) the equality

$$B[V^0, t, x_1, \dots, x_n, u_1^0(t, x_1, \dots, x_n), \dots, u_r^0(t, x_1, \dots, x_n)] = 0, \quad (20)$$

holds,
3) whichever the numbers u_j are, the following inequality holds

$$B[V^0, t, x_1, \dots, x_n, u_1, \dots, u_r] \geq 0,$$

then the functions $u_j^0(t, x_1, \dots, x_n)$ are the solution of the problem of optimal stabilization. Furthermore, the equality holds:

$$J = \int_{t_0}^{\infty} \omega(t, x_1^0(t), \dots, x_n^0(t), u_1^0(t), \dots, u_r^0(t))dt \quad (21)$$

$$= \min \int_{t_0}^{\infty} \omega(t, x_1(t), \dots, x_n(t), u_1(t), \dots, u_r(t))dt = V^0(t_0, x_1(t_0), \dots, x_n(t_0)).$$

Consider now the problem of stabilization for one of the families of steady-state motions (12):

$$\{s_1^0 = -\frac{6\lambda_2^{3/2}}{\alpha}, s_2^0 = 0, s_3^0 = 0, r_1^0 = 0, r_2^0 = 0, r_3^0 = 0\}, \quad (22)$$

obtained under the condition imposed on λ_i: $\lambda_0 = 0, \lambda_1 = 0, \lambda_3 = -1/(4\alpha^2\lambda_2^2)$.

The family of solutions (22) are unstable in the first approximation [1], because among the roots of the characteristic equation constructed for the linearized equations of system (3) in the neighbourhood of (22) there are zero multiple roots. The Jordan form of the system matrix with nonzero over-diagonal elements corresponds to these roots. This means that the family of steady-state motions under scrutiny is unstable in the first approximation.

Consider the problem of optimal stabilization in the first approximation for the family of solutions (22) under the condition of incomplete control u_1, u_2, u_3, u_4, u_5.

Equations of the 1st approximation with control for the system (3), which are obtained in the neighborhood of (22), write:

$$\dot{z}_1 = u_1, \ \dot{z}_2 = \frac{6\lambda_2^{3/2}}{\alpha} z_3 + 6\lambda_2^{3/2} z_4 + u_2, \ \dot{z}_3 = u_3,$$

$$\dot{z}_4 = u_4, \ \dot{z}_5 = u_5 - \frac{6\lambda_2^{3/2}}{\alpha} z_6, \ \dot{z}_6 = \frac{6\lambda_2^{3/2}}{\alpha} z_5, \quad (23)$$

where $z_1 = s_1 - s_1^0, z_2 = s_2 - s_2^0, z_3 = s_3 - s_3^0$, $z_4 = r_1 - r_1^0, z_5 = r_2 - r_2^0, z_6 = r_3 - r_3^0$ are deviations.

The system (23) is controlled, what follows from direct verification of the Kalman criterion [8].

Let us choose the following quadratic form in the capacity of the function ω (17)

$$\omega = \frac{1}{2}(u_1^2 + u_2^2 + u_3^2 + u_4^2 + u_5^2) + a_1u_1z_1 + a_2u_2z_2 + a_3u_3z_3 + a_4u_4z_4 + a_5u_5z_5$$
$$+a_6u_5z_6 + \frac{1}{2}(d_1^2z_1^2 + d_2^2z_2^2 + d_3^2z_3^2 + d_4^2z_4^2 + d_6^2z_6^2) + b_2z_2z_3 + b_1z_2z_4, \qquad (24)$$

and in the capacity of the Lyapunov function –

$$V = \frac{1}{2}(c_1z_1^2 + c_2z_2^2 + c_3z_3^2 + c_4z_4^2 + c_5z_5^2 + c_6z_6^2) + c_7z_5z_6, \qquad (25)$$

where a_i, c_i $(i = 1, \dots, 6)$, c_7, b_1, b_2, d_1, d_2, d_3, d_4, d_6 are arbitrary constants to be defined.

The expression for B (18) will have the form:

$$B = \frac{1}{2}(u_1^2 + u_2^2 + u_3^2 + u_4^2 + u_5^2) + \frac{1}{2}(d_1^2z_1^2 + d_2^2z_2^2 + d_3^2z_3^2 + d_4^2z_4^2 + d_6^2z_6^2) + a_1u_1z_1$$
$$+a_2u_2z_2 + a_3u_3z_3 + a_4u_4z_4 + a_5u_5z_5 + a_6u_5z_6 + c_1u_1z_1 + c_2u_2z_2 + c_3u_3z_3$$
$$+c_4u_4z_4 + c_5u_5z_5 + c_7u_5z_6 + b_1z_2z_4 + b_2z_2z_3 + \frac{6c_2\lambda_2^{3/2}}{\alpha}z_2z_3 + 6c_2\lambda_2^{3/2}z_2z_4$$
$$+\frac{6c_7\lambda_2^{3/2}}{\alpha}z_5^2 + \frac{6(c_6 - c_5)\lambda_2^{3/2}}{\alpha}z_5z_6 - \frac{6c_7\lambda_2^{3/2}}{\alpha}z_6^2. \qquad (26)$$

The problem implies obtaining the controls $u_1^0, u_2^0, u_3^0, u_4^0, u_5^0$, which would satisfy conditions (19)-(21) of the theorem of optimal stabilization.

From the equations

$$\frac{\partial B}{\partial u_i} = 0, \qquad (i = 1, \dots, 5)$$

it is possible to find out expressions which define the structure of the controls $u_1^0, u_2^0, u_3^0, u_4^0, u_5^0$:

$$u_1^0 = -(a_1 + c_1)z_1,\ u_2^0 = -(a_2 + c_2)z_2,\ u_3^0 = -(a_3 + c_3)z_3,$$
$$u_4^0 = -(a_4 + c_4)z_4,\ u_5^0 = -(a_5 + c_5)z_5 - (a_6 + c_7)z_6. \qquad (27)$$

After the substitution of (27) into (26), the latter writes:

$$B = \frac{1}{2}(d_1^2 - (a_1 + c_1)^2)z_1^2 + \frac{1}{2}(d_2^2 - (a_2 + c_2)^2)z_2^2 + \frac{1}{2}(d_3^2 - (a_3 + c_3)^2)z_3^2$$
$$+\frac{1}{2}(d_4^2 - (a_4 + c_4)^2)z_4^2 + (\frac{6c_7\lambda_2^{3/2}}{\alpha} - \frac{1}{2}(a_5 + c_5)^2)z_5^2 + \frac{1}{2\alpha}(\alpha(d_6^2 \qquad (28)$$
$$-(a_6 + c_7)^2) - 12c_7\lambda_2^{3/2})z_6^2 + (b_2 + \frac{6c_2\lambda_2^{3/2}}{\alpha})z_2z_3 + (b_1 + 6c_2\lambda_2^{3/2})z_2z_4$$
$$+(\frac{6(c_6 - c_5)\lambda_2^{3/2}}{\alpha} - (a_5 + c_5)(a_6 + c_7))z_5z_6 = 0.$$

Satisfaction of equality (28) for any values of z_j $(j = 1, \dots, 6)$ necessitates that all the coefficients of $z_i z_j$ $(j = 1, \dots, 6; i = 1, \dots, 6)$ turn into zero. Having equated these coefficients in (28) to zero, we obtain a system of algebraic equations needed for determining the a_i, c_i $(i = 1, \dots, 6)$, $c_7, b_1, b_2, d_1, d_2, d_3, d_4, d_6$. In the capacity of one of its solutions this system has the following one:

$$b_1 = -6c_2\lambda_2^{3/2},\ b_2 = -\frac{6c_2\lambda_2^{3/2}}{\alpha},\ a_1 = d_1 - c_1,\ a_2 = d_2 - c_2,\ a_3 = d_3 - c_3, \quad (29)$$
$$a_4 = d_4 - c_4,\ a_5 = \frac{2\sqrt{3}\sqrt{c_7}\lambda_2^{3/4}}{\sqrt{\alpha}} - c_6,\ a_6 = -c_7,\ c_5 = c_6,\ d_6 = -\frac{2\sqrt{3}\sqrt{c_7}\lambda_2^{3/4}}{\sqrt{\alpha}}.$$

Having substituted (29) into (27), (24), and (25), we can obtain the desired expressions for the controls $u_1^0, u_2^0, u_3^0,\ u_4^0, u_5^0$ and for the functions $\omega,\ V$:

$$u_1^0 = -d_1 z_1, u_2^0 = -d_2 z_2, u_3^0 = -d_3 z_3, u_4^0 = -d_4 z_4, u_5^0 = -\frac{2\sqrt{3}\sqrt{c_7}\lambda_2^{3/4}}{\sqrt{\alpha}} z_5. \quad (30)$$

$$\begin{aligned}\omega &= c_1 d_1 z_1^2 + c_2 d_2 z_2^2 - \frac{6c_2\lambda_2^{3/2}}{\alpha} z_2 z_3 + c_3 d_3 z_3^2 - 6c_2\lambda_2^{3/2} z_2 z_4 + c_4 d_4 z_4^2 \\ &+ \frac{2\sqrt{3}\sqrt{\alpha}c_6\sqrt{c_7}\lambda_2^{3/4} - 6c_7\lambda_2^{3/2}}{\alpha} z_5^2 + \frac{2\sqrt{3}c_7^{3/2}\lambda_2^{3/4}}{\sqrt{\alpha}} z_5 z_6 + \frac{6c_7\lambda_2^{3/2}}{\alpha} z_6^2, \\ V &= \frac{1}{2}(c_1 z_1^2 + c_2 z_2^2 + c_3 z_3^2 + c_4 z_4^2 + c_6 z_5^2 + c_6 z_6^2) + c_7 z_5 z_6.\end{aligned}$$

The functions ω and V are positive definite when the following inequalities hold:

$$\begin{aligned}&\alpha > 0 \vee \lambda_2 > 0 \\ &\vee c_1 > 0 \vee c_2 > 0 \vee c_3 > 0 \vee c_4 > 0 \vee c_7 = \frac{12\lambda_2^{3/2}}{\alpha} \vee c_6 > \frac{12\lambda_2^{3/2}}{\alpha} \\ &\vee d_1 > 0 \vee d_2 > 0 \vee d_3 > \frac{9c_2\lambda_2^3}{\alpha^2 c_3 d_2} \vee d_4 > \frac{9\alpha^2 c_2 c_3 d_3 \lambda_2^2}{\alpha^2 c_3 c_4 d_2 d_3 - 9c_2 c_4 \lambda_2^3}.\end{aligned} \quad (31)$$

The latter follows from the Sylvester conditions (necessary and sufficient conditions of sign-definiteness for quadratic forms) obtained for these functions.

Substitution of the control (30) into (23) shows that, when conditions (31) hold, the characteristic equation of the system (23) has the roots only with negative real parts. Consequently, we have obtained a solution for the problem of optimal stabilization of steady-state motions (22).

6 Conclusion

Some results of analysis of Kirchhoff equations, which describe the motion of a rigid body in the ideal incompressible fluid, have been discussed. Steady-state motions, invariant manifolds of steady-state motions have been found out for these equations. Qualitative analysis of properties of obtained motions has been conducted in the aspect of stability and stabilization of motion. The methods of investigation used are based on classical results obtained by Lyapunov [1]. The CAS “Mathematica”, “Maple”, and a software [2] were used as the analytical tools. Lyapunov's sufficient stability conditions have been derived for a set of obtained steady-state motions, which, concerning the problem under scrutiny, are normally helical motions or families of such motions of the body. As far as cases of unstable motion are concerned, we have solved the problem of optimal stabilization with respect to the 1st approximation equations.

References

1. Lyapunov, A.M.: The general problem of stability of motion. Collected Works. USSR Acad. Sci. Publ., Moscow–Leningrad **2** (1956)
2. Irtegov, V.D., Titorenko, T.N.: Computer algebra and investigation of invariant manifolds of mechanical systems. Mathematics and Computers in Simulation. Special Issue: Applications of Computer Algebra in Science, Engineering, Simulation and Special Software. North-Holland, Elsevier **(1-2)**67 (2004) 99–109
3. Irtegov, V., Titorenko, T.: On stability of body motions in fluid. In: Proc. Seventh Workshop on Computer Algebra in Scientific Computing. Munich Techn. Univ. (2004) 261–268
4. Dolzhansky, F.V.: On hydrodynamical interpretation of equations of rigid body motion. Izvestiya of Russian Academy of Sciences. Physics of Atmosphere and Ocean. **2**(13) (1977) 201–204
5. Oparina, E.I., Troshkin, O.V.: Stability of Kolmogorov's flow in a channel with solid walls. Russian Math. Surveys **4**(398) (2004) 487–491
6. Oden, M.: Rotating Tops: A Course Integrable Systems. Izhevsk, Udmurtiya univ. (1999)
7. Lyapunov, A.M.: On Permanent Helical Motions of a Rigid Body in Fluid. Collected Works. USSR Acad. Sci. Publ., Moscow–Leningrad **1**(1954) 276–319
8. Malkin, I.G.: Stability Theory. Nauka Publ., Moscow (1966)
9. Sokolov, V.V.: A new integrable case for Kirchhoff equations. In: Theoretical and Mathematical Physics. Nauka, Moscow **1**(129) (2001) 31–37
10. Borisov, A.V., Mamayev, I.S., Sokolov, V.V.: A new integrable case on $so(4)$. Russian Math. Surveys **5**(381) (2001) 614–615
11. Rumyantsev, V.V.: Comparison of three methods of constructing Lyapunov functions. Applied Mathematics and Mechanics **6**(59) (1995) 916–921
12. Irtegov, V.D.: Invariant Manifolds of Steady-State Motions and Their Stability. Nauka Publ., Novosibirsk (1985)

On Compatibility of Discrete Relations

Vladimir V. Kornyak

Laboratory of Information Technologies,
Joint Institute for Nuclear Research,
141980 Dubna, Russia
kornyak@jinr.ru

Abstract. An approach to compatibility analysis of systems of discrete relations is proposed. Unlike the Gröbner basis technique, the proposed scheme is not based on the polynomial ring structure. It uses more primitive set-theoretic and topological concepts and constructions. We illustrate the approach by application to some two-state cellular automata. In the two-state case the Gröbner basis method is also applicable, and we compare both approaches.

1 Introduction

A typical example of a system of discrete relations is a cellular automaton. Cellular automata are used successfully in a large number of applications.[1] Furthermore, the concept of cellular automaton can be generalized, and we consider the following extension of the standard notion of a cellular automaton:

1. Instead of regular uniform lattice representing the space and time in a cellular automaton, we consider more general *abstract simplicial complex* $K = (X, \Delta)$ (see, e.g., [2]). Here $X = \{x_0, x_1, \ldots\}$ is a finite (or countably infinite) set of *points*; Δ is a collection of subsets of X such that (a) for all $x_i \in X$, $\{x_i\} \in \Delta$; (b) if $\tau \subseteq \delta \in \Delta$, then $\tau \in \Delta$.
   The sets $\{x_i\}$ are called *vertices*. We say $\delta \in \Delta$ is a $k-$*simplex* of *dimension* k if $|\delta| = k+1$, i.e., $\dim \delta = |\delta| - 1$. The *dimension of complex* K is defined as the maximum dimension of its constituent simplices $\dim K = \max\limits_{\delta \in \Delta} \dim \delta$.
   If $\tau \subseteq \delta$, τ is called a *face* of δ. Since any face of a simplex is also a simplex, the topological structure of the complex K, i.e., the set Δ is uniquely determined by the set of *maximal simplices* under inclusion.
   One of the advantages of simplicial complexes over regular lattices is their applicability to models with dynamically emerging and evolving rather than pre-existing space-time structure.

[1] Comparing expressiveness of cellular automata and differential equations, T. Toffoli writes [1]: "Today, it is clear that we can do all that differential equations can do, and more, because it is differential equations that are the poor man's cellular automata — not the other way around!"

2. The dynamics of a cellular automaton is determined by a *local rule*
$$x_{i_k} = f\left(x_{i_0}, \ldots, x_{i_{k-1}}\right). \tag{1}$$
In this formula $x_{i_0}, \ldots, x_{i_k} \in X$ are interpreted as discrete variables taking values in a finite set of states S canonically represented as
$$S = \{0, \ldots, q-1\}.$$
The set of points $\left\{x_{i_0}, \ldots, x_{i_{k-1}}\right\}$ is called the *neighborhood*. The point x_{i_k} is considered as the "next time step" match of some point, say $x_{i_{k-1}}$, from the neighborhood.
A natural generalization is to replace function (1) by a *relation* on the set $\left\{x_{i_0}, \ldots, x_{i_k}\right\}$. In this context, local rule (1) is a special case of relation. Relations like (1) are called *functional relations*. They are too restrictive in many applications. In particular, they violate in most cases the symmetry among points $x_{i_0}, \ldots, x_{i_k}$. Furthermore, we will see below that the functional relations, as a rule, have non-functional consequences.

We can formulate some natural problems concerning the above structures:

1. *Construction of consequences.* Given a relation R^{δ} on a set of points δ, construct non-trivial relations R^{τ} on subsets $\tau \subseteq \delta$, such that $R^{\delta} \Rightarrow R^{\tau}$.
2. *Extension of relation.* Given a relation R^{τ} on a subset $\tau \subseteq \delta$, extend it to relation R^{δ} on the superset δ.
3. *Decomposition of relation.* Given a relation R^{δ} on a set δ, decompose R^{δ} into combination of relations on subsets of δ.
4. *Compatibility problem.* Given a collection of relations $\left\{R^{\delta_1}, \ldots, R^{\delta_n}\right\}$ defined on sets $\{\delta_1, \ldots, \delta_n\}$, construct relation $R^{\cup_{i=1}^{n}\delta_i}$ on the union $\bigcup_{i=1}^{n}\delta_i$, such that $R^{\cup_{i=1}^{n}\delta_i}$ is compatible with the initial relations.
5. *Imposing topological structure.* Given a relation R^{X} on a set X, endow X with a structure of simplicial complex consistent with the decomposition of the relation.

If the number of states is a power of a prime, i.e., $q = p^n$, we can always[2] represent any relation over k points $\{x_1, \ldots, x_k\}$ by the set of zeros of some polynomial from the ring $\mathbb{F}_q[x_1, \ldots, x_k]$ and study the compatibility problem by the standard Gröbner basis methods. It would be instructive to look at the compatibility problem from the set-theoretic point of view cleared of the ring structure influence.

An example from fundamental physics is the *holographic principle* proposed by G. 't Hooft and developed by many authors (see [4,5]). According to 't Hooft the combination of quantum mechanics and gravity implies that the world at the Planck scale can be described by a three-dimensional discrete lattice theory with a spacing of the Planck length order. Moreover, a full description of events on

[2] Due to the functional completeness of polynomials over $\mathbb{F}_q$ (see [3]) any function mapping k elements of $\mathbb{F}_q$ into $\mathbb{F}_q$ can be realized by a polynomial.

the three-dimensional lattice can be derived from a set of Boolean data (one bit per Planck area) on a two-dimensional lattice at the spatial (evolving with time) boundaries of the world. The transfer of data from two to three dimensions is performed in accordance with some local relations (constraints or laws) defined on plaquettes of the lattice. Since the data on points of the three-dimensional lattice are overdetermined, the control of compatibility of relations is necessary. Large number of constraints compared to the freedom one has in constructing models is one of the reasons why no completely consistent mathematical models describing physics at the Planck scale have been found so far.

2 Basic Definitions and Constructions

The definition of *abstract* k-simplex as a set of $k+1$ points is motivated by the fact that $k+1$ points generically embedded in Euclidean space of sufficiently high dimension determine k-dimensional convex polyhedron. The abstract combinatorial topology only cares about how the simplices are connected, and not how they can be placed within whatever spaces.[3] We need to consider also k-point sets which we call *k-sets*. Notice that k-sets may or may not be $(k-1)$-simplices.

A relation is defined as a subset of a Cartesian product $S \times \cdots \times S$ of the set of states. Dealing with the system of relations determined over different sets of points we should indicate the correspondence between points and dimensions of the hypercube $S \times \cdots \times S$. The notation $S^{\{x_i\}}$ specifies the set S as a set of values for the point x_i. For the k-set $\delta = \{x_1, \dots, x_k\}$ we denote $S^\delta \equiv S^{\{x_1\}} \times \cdots \times S^{\{x_k\}}$.

A ***relation*** R^δ over a k-set $\delta = \{x_1, \dots, x_k\}$ is any subset of the hypercube S^δ, i.e., $R^\delta \subseteq S^\delta$. We call the set δ *domain* of the relation R^δ. The relations $\emptyset^\delta$ and S^δ are called *empty* and *trivial*, respectively.

Given a set of points δ, its subset $\tau \subseteq \delta$ and relation R^τ over the subset τ, we define ***extension*** of R^τ as the relation

$$R^\delta = R^\tau \times S^{\delta \setminus \tau}.$$

The procedure of extension allows one to extend relations $R^{\delta_1}, \dots, R^{\delta_m}$ defined on different domains to the common domain, i.e., the union $\delta_1 \cup \cdots \cup \delta_m$.

Now we can construct the ***compatibility condition*** of the system of relations $R^{\delta_1}, \dots, R^{\delta_m}$. Naturally this is intersection of extensions of the relations to the common domain

$$R^\delta = \bigcap_{i=1}^{m} \left(R^{\delta_i} \times S^{\delta \setminus \delta_i} \right), \text{ where } \delta = \bigcup_{i=1}^{m} \delta_i.$$

We call the compatibility condition R^δ the ***base relation*** of the system of relations $R^{\delta_1}, \dots, R^{\delta_m}$. If the base relation is empty, the relations $R^{\delta_1}, \dots, R^{\delta_m}$ are *incompatible.* Note that in the case $q = p^n$ the compatibility condition can

[3] There are mathematical structures of non-geometric origin, like *hypergraphs* or *block designs*, closely related conceptually to the abstract simplicial complexes.

be represented by a single polynomial, in contrast to the Gröbner basis approach (of course, the main aim of the Gröbner basis computation — construction of basis of polynomial ideal — is out of the question).

A relation Q^δ is a *consequence* of relation R^δ, if $R^\delta \subseteq Q^\delta \subseteq S^\delta$, i.e., Q^δ is any superset of R^δ. Any relation can be represented in many ways by intersections of different sets of its consequences:

$$R^\delta = Q^{\tau_1} \cap \cdots \cap Q^{\tau_r}.$$

We call such representations *decompositions.*

In the polynomial case $q = p^n$, any possible Gröbner basis of polynomials representing the relations $R^{\delta_1}, \ldots, R^{\delta_m}$ corresponds to some decomposition of the base relation R^δ of the system $R^{\delta_1}, \ldots, R^{\delta_m}$. However, the decomposition implied by a Gröbner basis may look accidental from our point of view and if $q \neq p^n$ such decomposition is impossible at all.

The total number of all consequences (including R^δ itself and the trivial relation S^δ) is, obviously,

$$2^{\left(q^k - |R^\delta|\right)}.$$

In our context it is natural to distinguish the consequences which are reduced to relations over smaller sets of points.

A nontrivial relation Q^τ is called ***proper consequence*** of relation R^δ if τ is a proper subset of δ, i.e., $\tau \subset \delta$, and relation $Q^\tau \times S^{\delta\setminus\tau}$ is consequence of R^δ.

There are relations without proper consequences and these relations are most fundamental for a given number of points k. We call such relations ***prime***.

If relation R^δ has proper consequences $R^{\delta_1}, \ldots, R^{\delta_m}$ we can construct its ***canonical decomposition***

$$R^\delta = PR^\delta \bigcap \left(\bigcap_{i=1}^{m} \left(R^{\delta_i} \times S^{\delta\setminus\delta_i} \right) \right), \qquad (2)$$

where the factor PR^δ, which we call the ***principal factor***, is defined as

$$PR^\delta = R^\delta \bigcup \left(S^\delta \setminus \bigcap_{i=1}^{m} \left(R^{\delta_i} \times S^{\delta\setminus\delta_i} \right) \right).$$

The principal factor is the relation of maximum "freedom", i.e., closest to the trivial relation but sufficient to restore R^δ in combination with the proper consequences.

If the principal factor in canonical decomposition (2) is trivial, then R^δ can be fully reduced to relations over smaller sets of points. We call a relation R^δ ***reducible***, if it can be represented in the form

$$R^\delta = \bigcap_{i=1}^{m} \left(R^{\delta_i} \times S^{\delta\setminus\delta_i} \right),$$

where all R^{δ_i} are proper consequences of R^{δ}. For brevity we will omit the trivial multipliers in intersections and write in the subsequent sections expressions like $\bigcap_{i=1}^{m} R^{\delta_i}$ instead of $\bigcap_{i=1}^{m} \left(R^{\delta_i} \times S^{\delta \setminus \delta_i}\right)$.

We see how to impose the structure of simplicial complex on an amorphous set of points $X = \{x_0, x_1, \ldots\}$ via a relation R^X. The maximal simplices of Δ must correspond to the irreducible components of the relation R^X. Now we can evolve — starting only with a set of points and a relation on it (in fact, we simply identify dimensions of the relation with the points) — the standard tools of the algebraic topology like homology, cohomology, etc.

We wrote a program in ***C*** implementing the above constructions and manipulations with them. Below we illustrate application of the program to analysis of Conway's Game of Life [6] and some of the Wolfram's elementary cellular automata [7].

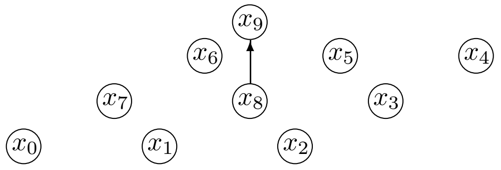
A few words are needed about computer implementation of relations. To specify a k-ary relation R^k we should mark its points within the k-dimensional hypercube S^k, i.e., define a *characteristic function* $\chi : S^k \rightarrow \{0, 1\}$, with $\chi(\mathbf{s}) = 1$ or 0 according as $\mathbf{s} \in R^k$ or $\mathbf{s} \notin R^k$. Here $\mathbf{s} = (s_0, s_1, \ldots, s_{k-1})$ is a point of the hypercube. The simplest way to implement the characteristic function is to enumerate all the q^k hypercube points in some standard, e.g., lexicographic order:

s_0	s_1	$\ldots$	s_{k-2}	s_{k-1}	i_{ord}
0	0	$\ldots$	0	0	0
1	0	$\ldots$	0	0	1
$\vdots$	$\vdots$	$\ldots$	$\vdots$	$\vdots$	$\vdots$
$q-2$	$q-1$	$\ldots$	$q-1$	$q-1$	q^k-2
$q-1$	$q-1$	$\ldots$	$q-1$	$q-1$	q^k-1

Then the relation can be represented by a string of q^k bits. We call this string *bit table* of relation. Symbolically $\mathrm{BitTable}\,[\,i_{ord}] := \left(\mathbf{s} \in R^k\right)$. Note that $\mathbf{s}$ is a ("little-endian") representation of the number i_{ord} in the base q. Most manipulations with relations are reduced to very efficient bitwise computer commands. Of course, symmetric or sparse (or, vice versa, dense) relations can be represented in a more economical way, but these are technical details of implementation.

3 Conway's Game of Life

The local rule of the cellular automaton ***Life*** is defined over the 10-set $\delta = \{x_0, \ldots, x_9\}$:



Here the point x_9 is the next time step of the point x_8. The state set S is $\{0, 1\}$. The local rule can be represented as a relation $R^{\delta}_{\mathit{Life}}$ on the 10-dimensional hypercube S^δ. By definition, the hypercube element belongs to the relation of the automaton ***Life***, i.e., $(x_0, \ldots, x_9) \in R^\delta_{\boldsymbol{Life}}$, in the following cases:

1. $\left(\sum_{i=0}^{7} x_i = 3\right) \wedge (x_9 = 1)$,
2. $\left(\sum_{i=0}^{7} x_i = 2\right) \wedge (x_8 = x_9)$,
3. $x_9 = 0$, if none of the above conditions holds.

The number of elements of $R^\delta_{\boldsymbol{Life}}$ is $\left|R^\delta_{\boldsymbol{Life}}\right| = 512$. The relation $R^\delta_{\boldsymbol{Life}}$, as is the case for any cellular automaton, is *functional*: the state of x_9 is uniquely determined by the states of other points. The state set $S = \{0, 1\}$ can be *additionally* endowed with the structure of the field $\mathbb{F}_2$. We accompany the below analysis of the structure of $R^\delta_{\boldsymbol{Life}}$ by description in terms of polynomials from $\mathbb{F}_2[x_0, \ldots, x_9]$. This is done only for illustrative purposes and for comparison with the Gröbner basis method. In fact, we transform the relations to polynomials only for output. This is done by computationally very cheap Lagrange interpolation generalized to the multivariate case. In the case $q = 2$, the polynomial which set of zeros corresponds to a relation is constructed uniquely. If $q = p^n > 2$, there is a freedom in the choice of nonzero values of constructed polynomial, and the same relation can be represented by many polynomials.

The polynomial representing $R^\delta_{\boldsymbol{Life}}$ takes the form

$$P_{\boldsymbol{Life}} = x_9 + x_8\left\{\sigma_7 + \sigma_6 + \sigma_3 + \sigma_2\right\} + \sigma_7 + \sigma_3, \tag{3}$$

where $\sigma_k \equiv \sigma_k(x_0, \ldots, x_7)$ is the kth *elementary symmetric polynomial* defined for n variables $x_0, \ldots, x_{n-1}$ by the formula:

$$\sigma_k(x_0, \ldots, x_{n-1}) = \sum_{0 \le i_0 < i_1 < \cdots < i_{k-1} < n} x_{i_0} x_{i_1} \cdots x_{i_{k-1}}.$$

The relation $R^\delta_{\boldsymbol{Life}}$ is reducible. It decomposes into two equivalence classes (with respect to the permutations of the points $x_0, \ldots, x_7$) of relations defined over 9 points:

1. Eight relations $R_1^{\delta \setminus \{x_i\}}, \quad 0 \le i \le 7$.
   Their polynomials $P_1^i(x_0, \ldots, \widehat{x_i}, \ldots, x_7, x_8, x_9)$ take the form

$$P_1^i = x_8 x_9\left\{\sigma_6^i + \sigma_5^i + \sigma_2^i + \sigma_1^i\right\} + x_9\left\{\sigma_6^i + \sigma_2^i + 1\right\} + x_8\left\{\sigma_7^i + \sigma_6^i + \sigma_3^i + \sigma_2^i\right\},$$

$$\sigma_k^i \equiv \sigma_k(x_0, \ldots, \widehat{x_i}, \ldots, x_7). \tag{4}$$

2. One relation $R_2^{\delta \setminus \{x_8\}}$ with polynomial $P_2^8(x_0, \ldots, x_7, x_9)$:

$$P_2^8 = x_9\left\{\sigma_7 + \sigma_6 + \sigma_3 + \sigma_2 + 1\right\} + \sigma_7 + \sigma_3, \qquad \sigma_k \equiv \sigma_k(x_0, \ldots, x_7). \tag{5}$$

The relation $R^{\delta}_{\boldsymbol{Life}}$ has the following decomposition

$$R^{\delta}_{\boldsymbol{Life}} = R_2^{\delta\setminus\{x_8\}} \bigcap \left(\bigcap_{k=0}^{6} R_1^{\delta\setminus\{x_{i_k}\}} \right), \tag{6}$$

where $(i_0, \ldots, i_6)$ are any 7 different indices from the set $(0, \ldots, 7)$.

We see that the rule of ***Life*** is defined on 8-dimensional space-time simplices. Of course, this interpretation is based on the concepts of the abstract combinatorial topology and differs from the native interpretation of the game of ***Life*** as a (2+1)-dimensional lattice structure.

The relations $R_1^{\delta\setminus\{x_i\}}$ and $R_2^{\delta\setminus\{x_8\}}$ are irreducible but not prime, i.e., they have proper consequences.

The relation $R_1^{\delta\setminus\{x_i\}}$ has two classes of 7-dimensional consequences:

1. Seven relations $R_{1.1}^{\delta\setminus\{x_i,x_j\}}$ with polynomials

$$\begin{aligned} P_{1.1}^{ij}\left(x_0, \ldots, \widehat{x_i}, \ldots, \widehat{x_j}, \ldots, x_7, x_8, x_9\right) = & \\ x_8x_9 \left\{ \sigma_6^{ij} + \sigma_5^{ij} + \sigma_4^{ij} + \sigma_3^{ij} + \sigma_2^{ij} + \sigma_1^{ij} + 1 \right\} & \\ + x_9 \left\{ \sigma_6^{ij} + \sigma_5^{ij} + \sigma_3^{ij} + \sigma_2^{ij} + \sigma_1^{ij} + 1 \right\}, & \qquad (7) \\ \sigma_k^{ij} \equiv \sigma_k \left(x_0, \ldots, \widehat{x_i}, \ldots, \widehat{x_j}, \ldots, x_7\right). & \end{aligned}$$

2. One relation $R_{1.2}^{\delta\setminus\{x_i,x_8\}}$ with polynomial

$$P_{1.2}^{i}\left(x_0, \ldots, \widehat{x_i}, \ldots, x_7, x_9\right) = x_9 \left\{ \sigma_7^{i} + \sigma_6^{i} + \sigma_5^{i} + \sigma_3^{i} + \sigma_2^{i} + \sigma_1^{i} + 1 \right\}. \tag{8}$$

The 8-dimensional relation $R_2^{\delta\setminus\{x_8\}}$ has one class of 7-dimensional consequences. This class contains 8 already obtained relations $R_{1.2}^{\delta\setminus\{x_i,x_8\}}$ with polynomials (8).

Continuing the process of construction of decompositions and proper consequences we come finally to the prime relations $R^{\delta_{i_0i_1i_2i_3}}$ defined over 4-simplices $\delta_{i_0i_1i_2i_3} = \{x_{i_0}, x_{i_1}, x_{i_2}, x_{i_3}, x_9\}$, where $i_k \in \{0, 1, \ldots, 7\}$ and $i_0 < i_1 < i_2 < i_3$. The polynomials of these relations take the form

$$P^{i_0,i_1,i_2,i_3} = x_9\sigma_4\left(x_{i_0}, x_{i_1}, x_{i_2}, x_{i_3}\right) \equiv x_9x_{i_0}x_{i_1}x_{i_2}x_{i_3}. \tag{9}$$

Substituting (9) in (4), (5), (7), and (8) (this is a *purely polynomial* simplification) we have finally the following polynomial form of the system of relations valid for the ***Life*** rule:

$$x_8x_9\left\{\sigma_2^{i} + \sigma_1^{i}\right\} + x_9\left\{\sigma_2^{i} + 1\right\} + x_8\left\{\sigma_7^{i} + \sigma_6^{i} + \sigma_3^{i} + \sigma_2^{i}\right\} = 0, \tag{10}$$

$$x_9\left\{\sigma_3 + \sigma_2 + 1\right\} + \sigma_7 + \sigma_3 = 0, \tag{11}$$

$$\left(x_8x_9 + x_9\right)\left\{\sigma_3^{ij} + \sigma_2^{ij} + \sigma_1^{ij} + 1\right\} = 0, \tag{12}$$

$$x_9\left\{\sigma_3^{i} + \sigma_2^{i} + \sigma_1^{i} + 1\right\} = 0, \tag{13}$$

$$x_9x_{i_0}x_{i_1}x_{i_2}x_{i_3} = 0. \tag{14}$$

Relations (14) have a simple interpretation: if the point x_9 is in the state 1, then at least one of any four points surrounding the center x_8 must be in the state 0.

The above analysis of the relation $R^{\delta}_{\boldsymbol{Life}}$ takes < 1 sec on a 1.8GHz AMD Athlon notebook with 960Mb.

To compute the Gröbner basis we must add to polynomial (3) ten polynomials

$$x_i^2 + x_i, \ i = 0, \ldots, 9, \tag{15}$$

expressing the relation $x^{p^n} = x$ valid for all elements of any finite field $\mathbb{F}_{p^n}$.

Computation of the Gröbner basis over $\mathbb{F}_2$ with the help of Maple 9 gives the following. Computation for the pure lexicographic order with the variable ordering $x_9 \succ x_8 \succ \cdots \succ x_0$ remains initial polynomial (3) unchanged, i.e., does not give any additional information. The pure lexicographic order with the variable ordering $x_0 \succ x_1 \succ \cdots \succ x_9$ gives relations (10)—(14) (modulo several polynomial reductions violating the symmetry of polynomials). The computation takes 1 h 22 min. Computation for the degree-reverse-lexicographic order also gives relations (10)—(14) (with the above reservation). The times are 51 min for the variable ordering $x_0 \succ x_1 \succ \cdots \succ x_9$, and 33 min for the ordering $x_9 \succ x_8 \succ \ldots \succ x_0$.

4 Elementary Cellular Automata

Simplest binary, nearest-neighbor, one-dimensional cellular automata were called *elementary cellular automata* by S. Wolfram, who has extensively studied their properties [7]. A large collection of results concerning these automata is presented in the Wolfram's online atlas [8]. In the exposition below we use Wolfram's notations and terminology. The elementary cellular automata are simpler than the ***Life***, and we may pay more attention to the topological aspects of our approach.

Local rules of the elementary cellular automata are defined on the 4-set $\delta = \{p, q, r, s\}$ which can be pictured by the icon

$$\begin{tikzcd}[column sep=small, row sep=small] \boxed{p} \arrow[dr, dash] & \boxed{q} \arrow[d, dash] & \boxed{r} \arrow[dl, dash] \\ & \boxed{s} & \end{tikzcd}$$

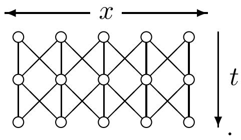
. A local rule is a binary function of the form $s = f(p, q, r)$. There are totally $2^{2^3} = 256$ local rules, each of which can be indexed with an 8-bit binary number.

Our computation with relations representing the local rules shows that the total number 256 of them is divided into 118 reducible and 138 irreducible relations. Only two of the irreducible relations appeared to be prime, namely, the rules 105 and 150[4] in Wolfram's numeration.[5]

We consider the elementary automata on a space-time lattice with integer coordinates (x, t), i.e., $x \in \mathbb{Z} = \{\ldots, -1, 0, 1, \ldots\}$ or $x \in \mathbb{Z}_m$ (spatial m-periodicity), $t \in \mathbb{Z}^* = \{0, 1, \ldots\}$. We denote a state of the point on the lattice by $u(x, t) \in S = \{0, 1\}$. Generally the points are connected as is shown on the 5×3 fragment of the lattice

[4] They are represented by the linear polynomial equations $p + q + r + s + 1 = 0$ and $p + q + r + s = 0$ for the rules 105 and 150, respectively.

[5] Wolfram prefers "big-endian" representation of binary numbers.



There are no horizontal ties due to the fundamental property of cellular automata — the states of points at a given temporal layer are independent.

Applying our approach we see that some automata with reducible local relations can be decomposed into automata on disjoint unions of subcomplexes:

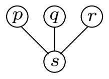
1. Two automata 0 and 255 are defined on disjoint union of vertices.
2. Six automata 15, 51, 85, 170, 204 and 240 are, in fact, disjoint collections of zero-dimensional automata. What we call *zero-dimensional automaton* is spatially zero-dimensional analog of the Wolfram's elementary automaton, i.e., a single cell evolving with time. There are, obviously, four such automata with local relations represented by the bit tables

$$\begin{aligned} &1100,\\ &0110,\\ &1001,\\ &0011. \end{aligned} \tag{16}$$

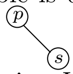
We call the automaton with bit table (16) *oscillating point* since its time evolution consists in periodic changing 0 by 1 and vice versa. It is easy to "integrate" these automata. Their general solutions are respectively

$$\begin{aligned} &u(t) = 0,\\ &u(t) = u(0) + t \mod 2, \quad \textit{oscillating point,}\\ &u(t) = u(0),\\ &u(t) = 1. \end{aligned} \tag{17}$$

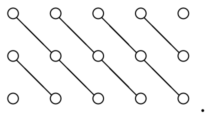
As an example consider the rule 15. The local relation is defined on the set



and its bit table is 0101010110101010. This relation is reduced to the relation on the face



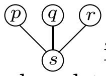
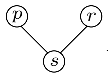
and its bit table 0110 coincides with bit table (16) of the oscillating point. We see that the automaton 15 decomposes into the union of identical zero-dimensional automata on the disconnected lattice



Using (17) we can write the general solution for the automaton 15

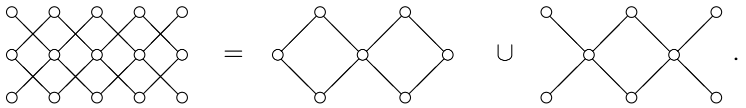
$$u(x,t) = u(x-t,0) + t \mod 2.$$

3. Ten automata 5, 10, 80, 90, 95, 160, 165, 175, 245, 250 are decomposed into two identical automata.

   As an example let us consider the rule 90. This automaton is distinguished as producing the fractal (of the topological dimension 1 and Hausdorff dimension $\ln 3/\ln 2 \approx 1.58$) known as the Sierpinski sieve, Sierpinski gasket, or Sierpinski triangle. Its local relation on the set  is represented by the bit table 1010010101011010. The relation is reduced to the relation on the face  with the bit table

$$10010110. \tag{18}$$

   From the structure of the domain of the reduced relation it is clear that the lattice decomposes into two identical independent lattices as is shown



   To find a general solution of the automaton 90 it is convenient to transform bit table (18) to an algebraic relation. It is the linear relation $s+p+r=0$ and the general solution of the automaton takes the form

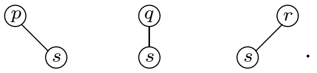
$$u(x,t)=\sum_{k=0}^{t}\binom{t}{k}u(x-t+2k,0) \mod 2.$$

In the above examples we have considered the automata with reducible relations. If a local relation is irreducible but has proper consequences we also, in some cases, can obtain a useful information.

For example, there are 64 automata[6] — both reducible and irreducible — having proper consequencies with the bit table

$$1101 \tag{19}$$

on one or two or three of the following faces

 (20)

The algebraic forms of relation (19) on faces (20) are

$$ps+s=0, \quad qs+s=0, \quad rs+s=0,$$

respectively.

[6] The full list of these automata in the Wolfram's numeration is 2, 4, 8, 10, 16, 32, 34, 40, 42, 48, 64, 72, 76, 80, 96, 112, 128, 130, 132, 136, 138, 140, 144, 160, 162, 168, 171, 174–176, 186, 187, 190–192, 196, 200, 205, 206, 208, 220, 222–224, 234–239, 241–254.

Relation (19) is non-functional. Nevertheless, it imposes a severe restriction on the behavior of the automata with such proper consequences. The peculiarities in the behavior are clear visible in the atlas [8], where many results of computations with different initial conditions are pictured. A typical pattern from this atlas is reproduced in Fig. 1, where several evolutions of the automaton 168 are presented. The local relation of the automaton 168 is $pqr + qr + pr + s = 0$. It has the proper consequence $rs + s = 0$. The black and white square cells in Fig. 1 correspond to 1's and 0's, respectively. Note also that the authors of Fig. 1 have used a spatially periodic condition. Their spacial variable is $x \in \mathbb{Z}_{30}$.

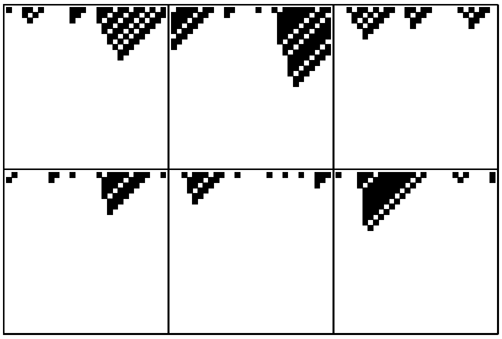


Fig. 1. Rule 168. Several random initial conditions

Relation (19) means that if, say r, as for rule 168, is in the state 1 then s may be in both states 0 or 1, but if the state of r is 0, then the state of s must be 0. Thus the corresponding diagonal or vertical may contain either only 1's, or finite number of initial 1's and then only 0's. The presence of a proper consequence of the form (19) simplifies essentially computation with such automata: after the first appearance of 0, one can set 0's on all points along the corresponding line.

In conclusion, let us present the results of analysis of the automata 30 and 110. These automata are of special interest. The automaton 30 demonstrates chaotic behavior and even used as the random number generator in ***Mathematica.*** The automaton 110 is, like a Turing machine, *universal*, i.e., it is capable of simulating any computational process, in particular, any other cellular automaton. The relations of both automata are irreducible but not prime.

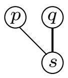
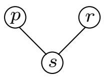
The relation of automaton 30 is

$$1001010101101010$$

or in the algebraic form

$$qr + s + r + q + p = 0.$$

It has two proper consequences:

face		
bit table	11011110	11011110
polynomial	$qs + pq + q$	$rs + pr + r$.

The principal factor is

$$1011111101111111 \quad \text{or} \quad qrs + pqr + rs + qs + pr + pq + s + p = 0.$$

The Gröbner basis of automaton 30 in the total degree and reverse lexicographic order is (omitting the trivial polynomials $p^2 + p,\ q^2 + q,\ r^2 + r,\ s^2 + s$)

$$\{qr + s + r + q + p,\ qs + pq + q,\ rs + pr + r\}\,.$$

We see that for the rule 30 the Gröbner basis polynomials coincide with ours.

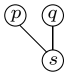
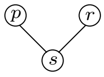
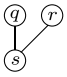
The relation of automaton 110 is

$$1100000100111110 \tag{21}$$

or in the polynomial form

$$pqr + qr + s + r + q = 0.$$

The relation has three proper consequences:

face			
bit table	11011111	11011111	10010111
polynomial	$pqs + qs + pq + q$	$prs + rs + pr + r$	$qrs + s + r + q$.

The principal factor is

$$1111111111111110 \quad \text{or} \quad pqrs = 0.$$

The Gröbner basis of automaton 110 contains different set of polynomials:

$$\{prs + rs + pr + r,\ qs + rs + r + q,\ qr + rs + s + q,\ pr + pq + ps\}\,.$$

The system of relations defined by the Gröbner basis is:

$$\begin{aligned}
R_1^{\{p,r,s\}} &= 11011111 = \{prs + rs + pr + r = 0\}\,,\\
R_2^{\{q,r,s\}} &= 10011111 = \{qs + rs + r + q = 0\}\,,\\
R_3^{\{q,r,s\}} &= 10110111 = \{qr + rs + s + q = 0\}\,,\\
R_4^{\{p,q,r,s\}} &= 1110101110111110 = \{pr + pq + ps = 0\}\,.
\end{aligned}$$

5 Conclusions

Let us summarize the main novelties of the paper.

- We have introduced a notion of a *system of discrete relations on an abstract simplicial complex.* Such a system can be interpreted as
  - a natural generalization of the notion of cellular automaton;
  - a set-theoretic analog of a system of polynomial equations.
- After introducing appropriate definitions, we have developed and implemented algorithms for
  - *compatibility analysis* of a system of discrete relations;
  - constructing *canonical decompositions* of discrete relations.
- We have proposed a regular way to impose *topology on an arbitrary discrete relation* via its canonical decomposition: identifying *dimensions* of the relation with *points* and *irreducible components* of the relation with *maximal simplices*, we define the structure of an abstract simplicial complex on the relation under consideration.
- Applying the above technique to some cellular automata — a special case of systems of discrete relations — we have obtained some new results. Most interesting of them, in our opinion, is demonstration of how the presence of non-trivial *proper consequences* may determine the global behavior of an automaton.

Acknowledgments

This work was supported in part by the grants 04-01-00784 from the Russian Foundation for Basic Research and 2339.2003.2 from the Russian Ministry of Industry, Science and Technologies.

References

1. Toffoli, T.: Occam, Turing, von Neumann, Jaynes: How much can you get for how little? (A conceptual introduction to cellular automata). In: *Automi Cellulari per la Ricerca e l'Industria, Rende (CS), Italy, September 29–30, 1994.*
2. Hilton, P. J., Wylie, S.: *Homology Theory: An Introduction to Algebraic Topology.* Cambridge University Press, New York (1960)
3. Lidl, R., Niederreiter, H.: *Finite Fields.* Encyclopedia of Math. and its Appl. **20**, Cambridge University Press, London (1997)
4. 't Hooft, G.: Dimensional Reduction in Quantum Gravity, Essay dedicated to Abdus Salam, Utrecht preprint THU-93/26, pp. 13; id., arXiv: gr-qc/9310026
5. Bousso, R.: The Holographic Principle. Rev.Mod.Phys. **74** (2002) 825–874. arXiv: hep-th/0203101
6. Gardner, M.: On Cellular Automata Self-reproduction, the Garden of Eden and the Game of 'Life'. Sci. Am., **224** (1971) 112–117
7. Wolfram, S.: *A New Kind of Science.* Wolfram Media, Inc (2002)
8. http://atlas.wolfram.com/01/01/

Construction of Two Level Orthogonal Arrays Via Solutions of Linear Systems

C. Koukouvinos and E. Lappas

Department of Mathematics, National Technical, University of Athens, Zografou 15773, Athens, Greece
ckoukouv@math.ntua.gr, elappas@central.ntua.gr

Abstract. In this paper we present a method that uses solutions of linear systems to construct all possible orthogonal arrays $OA(n, q, 2, 2^+)$, when $3 \leq q \leq 6$. A note on its complexity is also presented.

1 Introduction

An *orthogonal array* OA(n,q,s,t) is an $n \times q$ array with entries from a set of s distinct symbols arranged so that, for any collection of t columns of the array, each of the s^t row vectors appears equally often. Thus we see that s^t divides n. We call n the number of rows in the orthogonal array, q the number of columns, s the number of levels in each column and t the strength of the array.

Orthogonal arrays are useful in various fields such as, Applied Statistics (design of experiments, off-line quality control, etc), Coding Theory, Cryptography and more. For applications in Coding theory and Cryptography we refer to [2,8,9], and for a more recent application in visual cryptographic schemes see [4]. Also for more on orthogonal arrays, we refer the interested reader to Hedayat, Sloane and Stufken [5].

In this paper we are interested in constructing two level orthogonal arrays OA(n,q,2,t) with strength $t \geq 2$. We code the levels of each column with -1 and $+1$ so the sense of strength $t \geq 2$ reduces to the following two necessary conditions:

(a) the inner product of any two distinct columns in the array must be zero,
(b) the sum of the elements in each column must be equal to zero, that is we have the same number of -1 and $+1$ in every column.

A wide class of such orthogonal arrays can be obtained by selecting columns from Hadamard matrices. An n-dimensional Hadamard matrix H is an $n \times n$ matrix with elements -1 and $+1$ such that $H^T H = nI$, in other words any two columns of the Hadamard matrix H are orthogonal. The method of selecting columns from a Hadamard matrix has two major disadvantages. First, for instance when $n = 32$ it is known that there are at least 30000 inequivalent matrices. This explodes the combinatorial complexity of searching all possible columns. Second disadvantage is that we do not know all the possible inequivalent Hadamard matrices. But even if we knew them for a specific order there exist some OA(n,q,2,t) that cannot be embedded into Hadamard matrices and can be constructed using the method that would be described in this paper. On the other hand, in some statistical applications, where orthogonal arrays are used as factorial designs, those that cannot be embedded into Hadamard matrices are superior to the designs obtained from Hadamard matrices when further evaluated by the criteria of model estimability and design efficiency, see [6].

In order to construct all OA(n,q,2,t) $t \geq 2$ we will propose a method that generates all $n \times (q+1)$ matrices with elements ± 1 for which

(i) any two columns are orthogonal and
(ii) their first column has all elements equal to $+1$.

These two conditions are equivalent to the two conditions (a) and (b) given previously. Of course we should remove the first column with all $+1$ to get the OA(n,q,2,t).

Two orthogonal arrays are said to be isomorphic if, one can be obtained from the other by a sequence of permutations of the columns, the rows and the levels of each column. The identification of the complete non isomorphic classes of orthogonal arrays with specific parameters n, q and s, is a difficult combinatorial problem which seems to have important impact in various fields of Discrete Mathematics, see [1,3,7]. In this paper we are not dealing with the problem of isomorphism. Our focus is to construct all possible orthogonal arrays.

2 Construction of the Linear Systems and Its Solutions

2.1 Construction for q=3

The possible rows of the $n \times 4$ matrix satisfying the condition (ii) appear as vectors $\underline{u}_1, \ldots, \underline{u}_8$

$$\begin{aligned}
\underline{u}_1 &= (\,1 \quad 1 \quad 1 \quad 1\,)\\
\underline{u}_2 &= (\,1 \quad 1 \quad 1 \; -1\,)\\
\underline{u}_3 &= (\,1 \quad 1 \; -1 \quad 1\,)\\
\underline{u}_4 &= (\,1 \quad 1 \; -1 \; -1\,)\\
\underline{u}_5 &= (\,1 \; -1 \quad 1 \quad 1\,)\\
\underline{u}_6 &= (\,1 \; -1 \quad 1 \; -1\,)\\
\underline{u}_7 &= (\,1 \; -1 \; -1 \quad 1\,)\\
\underline{u}_8 &= (\,1 \; -1 \; -1 \; -1\,)
\end{aligned}$$

We denote with u_i the number of times the row vector $\underline{u}_i$ appears in the $n \times 4$ matrix. Since any two columns must be orthogonal we can write $\binom{4}{2} = 6$ equations, also $\sum_{i=1}^{8} u_i = n$. The system consists of 7 equations with 8 unknowns

$$\begin{aligned}
u_1 + u_2 + u_3 + u_4 - u_5 - u_6 - u_7 - u_8 &= 0\\
u_1 + u_2 - u_3 - u_4 + u_5 + u_6 - u_7 - u_8 &= 0\\
u_1 - u_2 + u_3 - u_4 + u_5 - u_6 + u_7 - u_8 &= 0
\end{aligned}$$

$$\begin{aligned}
u_1 + u_2 - u_3 - u_4 - u_5 - u_6 + u_7 + u_8 &= 0 \\
u_1 - u_2 + u_3 - u_4 - u_5 + u_6 - u_7 + u_8 &= 0 \\
u_1 - u_2 - u_3 + u_4 + u_5 - u_6 - u_7 + u_8 &= 0 \\
u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_7 + u_8 &= n
\end{aligned}$$

and has parametric solution

$$\begin{aligned}
u_1 &= n/4 - u_8 \\
u_2 &= u_8 \\
u_3 &= u_8 \\
u_4 &= n/4 - u_8 \\
u_5 &= u_8 \\
u_6 &= n/4 - u_8 \\
u_7 &= n/4 - u_8 \\
u_8 &= u_8
\end{aligned}$$

2.2 Construction for q=4

The possible rows of the $n \times 5$ matrix satisfying the condition (ii) appear as vectors $\underline{u}_1, \ldots, \underline{u}_{16}$

$$\begin{aligned}
\underline{u}_1 &= (\,1 \quad 1 \quad 1 \quad 1 \quad 1) \\
\underline{u}_2 &= (\,1 \quad 1 \quad 1 \quad 1 \, -1) \\
\underline{u}_3 &= (\,1 \quad 1 \quad 1 \, -1 \quad 1) \\
\underline{u}_4 &= (\,1 \quad 1 \quad 1 \, -1 \, -1) \\
\underline{u}_5 &= (\,1 \quad 1 \, -1 \quad 1 \quad 1) \\
\underline{u}_6 &= (\,1 \quad 1 \, -1 \quad 1 \, -1) \\
\underline{u}_7 &= (\,1 \quad 1 \, -1 \, -1 \quad 1) \\
\underline{u}_8 &= (\,1 \quad 1 \, -1 \, -1 \, -1) \\
\underline{u}_9 &= (\,1 \, -1 \quad 1 \quad 1 \quad 1) \\
\underline{u}_{10} &= (\,1 \, -1 \quad 1 \quad 1 \, -1) \\
\underline{u}_{11} &= (\,1 \, -1 \quad 1 \, -1 \quad 1) \\
\underline{u}_{12} &= (\,1 \, -1 \quad 1 \, -1 \, -1) \\
\underline{u}_{13} &= (\,1 \, -1 \, -1 \quad 1 \quad 1) \\
\underline{u}_{14} &= (\,1 \, -1 \, -1 \quad 1 \, -1) \\
\underline{u}_{15} &= (\,1 \, -1 \, -1 \, -1 \quad 1) \\
\underline{u}_{16} &= (\,1 \, -1 \, -1 \, -1 \, -1)
\end{aligned}$$

We denote with u_i the number of times the row vector $\underline{u}_i$ appears in the $n \times 5$ matrix. Since any two columns must be orthogonal we can write $\binom{5}{2} = 10$ equations, also $\sum_{i=1}^{16} u_i = n$. The system consists of 11 equations with 16 unknowns, and has parametric solution

$$\begin{aligned}
u_1 &= n/2 - u_8 - u_{12} - u_{14} - u_{15} - 3u_{16}\\
u_2 &= n/4 + u_8 + u_{12} + u_{14} + 2u_{16}\\
u_3 &= n/4 + u_8 + u_{12} + u_{15} + 2u_{16}\\
u_4 &= n/4 - u_8 - u_{12} - u_{16}\\
u_5 &= n/4 + u_8 + u_{14} + u_{15} + 2u_{16}\\
u_6 &= n/4 - u_8 - u_{14} - u_{16}\\
u_7 &= n/4 - u_8 - u_{15} - u_{16}\\
u_9 &= n/4 + u_{12} + u_{14} + u_{15} + 2u_{16}\\
u_{10} &= n/4 - u_{12} - u_{14} - u_{16}\\
u_{11} &= n/4 - u_{12} - u_{15} - u_{16}\\
u_{13} &= n/4 - u_{14} - u_{15} - u_{16}
\end{aligned}$$

where $u_8, u_{12}, u_{14}, u_{15}, u_{16}$ are the parameters of the solution.

2.3 Construction for q=5

Using the same techniques as previously we need to create $n \times 6$ matrices. The number of possible rows satisfying condition (ii) are 32. From orthogonality we can create $\binom{6}{2} = 15$ equations also $\sum_{i=1}^{32} u_i = n$. The system consists of 16 equations with 32 unknowns, and has parametric solution

$$\begin{aligned}
u_1 &= -n/2 - u_8 - u_{12} - u_{14} - u_{15} - 3u_{16} - u_{20} - u_{22} - u_{23} - 3u_{24} + 6u_{25}\\
&\quad +5u_{26} + 5u_{27} + 3u_{28} + 5u_{29} + 3u_{30} + 3u_{31}\\
u_2 &= n/4 + u_8 + u_{12} + u_{14} + 2u_{16} + u_{20} + u_{22} + 2u_{24} - 3u_{25} - 2u_{26} - 3u_{27}\\
&\quad -u_{28} - 3u_{29} - u_{30} - 3u_{31}\\
u_3 &= n/4 + u_8 + u_{12} + u_{15} + 2u_{16} + u_{20} + u_{23} + 2u_{24} - 3u_{25} - 3u_{26} - 2u_{27}\\
&\quad -u_{28} - 3u_{29} - 3u_{30} - u_{31}\\
u_4 &= -u_8 - u_{12} - u_{16} - u_{20} - u_{24} + u_{25} + u_{26} + u_{27} + u_{29} + u_{30} + u_{31}\\
u_5 &= n/4 + u_8 + u_{14} + u_{15} + 2u_{16} + u_{22} + u_{23} + 2u_{24} - 3u_{25} - 3u_{26} - 3u_{27}\\
&\quad -3u_{28} - 2u_{29} - u_{30} - u_{31}\\
u_6 &= -u_8 - u_{14} - u_{16} - u_{22} - u_{24} + u_{25} + u_{26} + u_{27} + u_{28} + u_{29} + u_{31}\\
u_7 &= -u_8 - u_{15} - u_{16} - u_{23} - u_{24} + u_{25} + u_{26} + u_{27} + u_{28} + u_{29} + u_{30}\\
u_9 &= n/4 + u_{12} + u_{14} + u_{15} + 2u_{16} - 3u_{25} - 2u_{26} - 2u_{27} - u_{28} - 2u_{29}\\
&\quad -u_{30} - u_{31}\\
u_{10} &= -u_{12} - u_{14} - u_{16} + u_{25} + u_{27} + u_{29} + u_{31}\\
u_{11} &= -u_{12} - u_{15} - u_{16} + u_{25} + u_{26} + u_{29} + u_{30}\\
u_{13} &= -u_{14} - u_{15} - u_{16} + u_{25} + u_{26} + u_{27} + u_{28}\\
u_{17} &= n/4 + u_{20} + u_{22} + u_{23} + 2u_{24} - 3u_{25} - 2u_{26} - 2u_{27} - u_{28} - 2u_{29}\\
&\quad -u_{30} - u_{31}
\end{aligned}$$

$$\begin{aligned}
u_{18} &= u_{29} + u_{31} - u_{20} - u_{24} + u_{25} + u_{27} - u_{22}\\
u_{19} &= u_{29} + u_{30} - u_{20} - u_{23} - u_{24} + u_{25} + u_{26}\\
u_{21} &= -u_{23} - u_{24} + u_{25} + u_{26} + u_{27} + u_{28} - u_{22}\\
u_{32} &= n/4 - u_{28} - u_{25} - u_{26} - u_{27} - u_{29} - u_{30} - u_{31}
\end{aligned}$$

where $u_8, u_{12}, u_{14}, u_{15}, u_{16}, u_{20}, u_{22}, \ldots, u_{31}$ are the parameters of the solution.

2.4 Construction for q=6

Using the same techniques as previously we need to create $n \times 7$ matrices. The number of possible rows satisfying condition (ii) are 64. From orthogonality we can create $\binom{7}{2} = 21$ equations also $\sum_{i=1}^{64} u_i = n$. The system consists of 22 equations with 64 unknowns, and has parametric solution

$$\begin{aligned}
u_1 &= 7n/4 - u_8 - u_{23} - u_{22} - 3u_{24} - u_{26} - u_{27} - u_{12} - u_{14} - u_{15} - 3u_{16} - u_{20}\\
&\quad -3u_{28} - u_{29} - 3u_{30} - 10u_{64} - 6u_{63} - 6u_{60} - 3u_{61} - 6u_{62} - 6u_{56} - u_{57}\\
&\quad -3u_{58} - 3u_{59} - 3u_{55} - u_{51} - 3u_{52} - u_{53} - 3u_{54} - u_{50} - u_{45} - 3u_{46}\\
&\quad -3u_{47} - 6u_{48} - u_{43} - 3u_{44} - 3u_{40} - u_{39} - u_{42} - 3u_{31} - 6u_{32} - u_{36} - u_{38}\\
u_2 &= -3n/4 + u_8 + u_{22} + 2u_{24} + u_{26} + u_{12} + u_{14} + 2u_{16} + u_{20} + 2u_{28} + 2u_{30}\\
&\quad +4u_{64} + 3u_{60} + 3u_{62} + 3u_{56} + 2u_{58} + 2u_{52} + 2u_{54} + u_{50} + 2u_{46} + 3u_{48}\\
&\quad +2u_{44} + 2u_{40} + u_{42} + 3u_{32} + u_{36} + u_{38}\\
u_3 &= -3n/4 + u_8 + u_{23} + 2u_{24} + u_{27} + u_{12} + u_{15} + 2u_{16} + u_{20} + 2u_{28} + 4u_{64}\\
&\quad +3u_{63} + 3u_{60} + 3u_{56} + 2u_{59} + 2u_{55} + u_{51} + 2u_{52} + 2u_{47} + 3u_{48} + u_{43}\\
&\quad +2u_{44} + 2u_{40} + u_{39} + 2u_{31} + 3u_{32} + u_{36}\\
u_4 &= n/4 - u_8 - u_{12} - u_{16} - u_{20} - u_{24} - u_{28} - u_{32} - u_{36} - u_{40} - u_{44}\\
&\quad -u_{48} - u_{52} - u_{56} - u_{60} - u_{64}\\
u_5 &= -3n/4 + u_8 + u_{23} + u_{22} + 2u_{24} + u_{14} + u_{15} + 2u_{16} + u_{29} + 2u_{30} + 4u_{64}\\
&\quad +3u_{63} + 2u_{61} + 3u_{62} + 3u_{56} + 2u_{55} + u_{53} + 2u_{54} + u_{45} + 2u_{46} + 2u_{47}\\
&\quad +3u_{48} + 2u_{40} + u_{39} + 2u_{31} + 3u_{32} + u_{38}\\
u_6 &= n/4 - u_8 - u_{14} - u_{16} - u_{22} - u_{24} - u_{30} - u_{32} - u_{38} - u_{40} - u_{46} - u_{48}\\
&\quad -u_{54} - u_{56} - u_{62} - u_{64}\\
u_7 &= n/4 - u_8 - u_{15} - u_{16} - u_{23} - u_{24} - u_{31} - u_{32} - u_{39} - u_{40} - u_{47} - u_{48}\\
&\quad -u_{55} - u_{56} - u_{63} - u_{64}\\
u_9 &= -3n/4 + u_{26} + u_{27} + u_{12} + u_{14} + u_{15} + 2u_{16} + 2u_{28} + u_{29} + 2u_{30} + 4u_{64}\\
&\quad +3u_{63} + 3u_{60} + 2u_{61} + 3u_{62} + u_{57} + 2u_{58} + 2u_{59} + u_{45} + 2u_{46} + 2u_{47}\\
&\quad +3u_{48} + u_{43} + 2u_{44} + u_{42} + 2u_{31} + 3u_{32}\\
u_{10} &= n/4 - u_{12} - u_{14} - u_{16} - u_{26} - u_{28} - u_{30} - u_{32} - u_{42} - u_{44} - u_{46} - u_{48}\\
&\quad -u_{58} - u_{60} - u_{62} - u_{64}
\end{aligned}$$

$$
\begin{aligned}
u_{11} &= n/4 - u_{12} - u_{15} - u_{16} - u_{27} - u_{28} - u_{31} - u_{32} - u_{43} - u_{44} - u_{47} - u_{48} \\
&\quad - u_{59} - u_{60} - u_{63} - u_{64} \\
u_{13} &= n/4 - u_{14} - u_{15} - u_{16} - u_{29} - u_{30} - u_{31} - u_{32} - u_{45} - u_{46} - u_{47} \\
&\quad - u_{48} - u_{48} - u_{61} - u_{62} - u_{63} - u_{64} \\
u_{17} &= -3n/4 + u_{23} + u_{22} + 2u_{24} + u_{26} + u_{27} + u_{20} + 2u_{28} + u_{29} + 2u_{30} + 4u_{64} \\
&\quad + 3u_{63} + 3u_{60} + 2u_{61} + 3u_{62} + 3u_{56} + u_{57} + 2u_{58} + 2u_{59} + 2u_{55} + u_{51} \\
&\quad + 2u_{52} + u_{53} + 2u_{54} + u_{50} + 2u_{31} + 3u_{32} \\
u_{18} &= n/4 - u_{60} - u_{20} - u_{22} - u_{24} - u_{26} - u_{28} - u_{30} - u_{32} - u_{50} - u_{52} - u_{54} \\
&\quad - u_{56} - u_{58} - u_{62} - u_{64} \\
u_{19} &= n/4 - u_{60} - u_{20} - u_{23} - u_{24} - u_{27} - u_{28} - u_{31} - u_{32} - u_{51} - u_{52} - u_{55} \\
&\quad - u_{56} - u_{59} - u_{63} - u_{64} \\
u_{21} &= n/4 - u_{62} - u_{22} - u_{23} - u_{24} - u_{29} - u_{30} - u_{31} - u_{32} - u_{53} - u_{54} - u_{55} \\
&\quad - u_{56} - u_{61} - u_{63} - u_{64} \\
u_{25} &= n/4 - u_{62} - u_{26} - u_{27} - u_{28} - u_{29} - u_{30} - u_{31} - u_{32} - u_{57} - u_{58} - u_{59} \\
&\quad - u_{60} - u_{61} - u_{63} - u_{64} \\
u_{33} &= -3n/4 + 4u_{64} + 3u_{63} + 3u_{60} + 2u_{61} + 3u_{62} + 3u_{56} + u_{57} + 2u_{58} + 2u_{59} \\
&\quad + 2u_{55} + u_{51} + 2u_{52} + u_{53} + 2u_{54} + u_{50} + u_{45} + 2u_{46} + 2u_{47} + 3u_{48} \\
&\quad + u_{43} + 2u_{44} + 2u_{40} + u_{39} + u_{42} + u_{36} + u_{38} \\
u_{34} &= n/4 - u_{60} - u_{36} - u_{38} - u_{40} - u_{42} - u_{44} - u_{46} - u_{48} - u_{50} - u_{52} - u_{54} \\
&\quad - u_{56} - u_{58} - u_{62} - u_{64} \\
u_{35} &= n/4 - u_{60} - u_{36} - u_{39} - u_{40} - u_{43} - u_{44} - u_{47} - u_{48} - u_{51} - u_{52} - u_{55} \\
&\quad - u_{56} - u_{59} - u_{63} - u_{64} \\
u_{37} &= n/4 - u_{62} - u_{38} - u_{39} - u_{40} - u_{45} - u_{46} - u_{47} - u_{48} - u_{53} - u_{54} - u_{55} \\
&\quad - u_{56} - u_{61} - u_{63} - u_{64} \\
u_{41} &= n/4 - u_{62} - u_{42} - u_{43} - u_{44} - u_{45} - u_{46} - u_{47} - u_{48} - u_{57} - u_{58} - u_{59} \\
&\quad - u_{60} - u_{61} - u_{63} - u_{64} \\
u_{49} &= n/4 - u_{62} - u_{50} - u_{51} - u_{52} - u_{53} - u_{54} - u_{55} - u_{56} - u_{57} - u_{58} - u_{59} \\
&\quad - u_{60} - u_{61} - u_{63} - u_{64}
\end{aligned}
$$

where $u_8, u_{12}, u_{14}, u_{15}, u_{16}, u_{20}, u_{22}, \ldots, u_{24}, u_{26}, \ldots, u_{32}, u_{36}, u_{38}, \ldots, u_{40}$, $u_{42}, \ldots, u_{48}, u_{50}, \ldots, u_{64}$ are the parameters of the solution.

3 Complexity and Results

A very helpful lemma for calculating and reducing the complexity of the values the parameters can take in our proposed method is the following.

Lemma 1. *(The Distribution Lemma) Let n be a multiple of 4, $n \geq 4$, and $\underline{u}_1 = (1,1,1)$, $\underline{u}_2 = (1,1,-1)$, $\underline{u}_3 = (1,-1,1)$, $\underline{u}_4 = (1,-1,-1)$. The $n \times 3$ matrix* *for which (i) it's first column are all +1's and (ii) it's columns are mutually orthogonal, has exactly $n/4$ rows of each $\underline{u}_i$, $i = 1, \ldots, 4$.*

Proof. Denote with u_i, $i = 1, \ldots, 4$ the number of times each $\underline{u}_i$ row appears in the $n \times 3$ matrix. Then by the order and the orthogonality we have:

$$\begin{aligned} u_1 + u_2 + u_3 + u_4 &= n \\ u_1 + u_2 - u_3 - u_4 &= 0 \\ u_1 - u_2 + u_3 - u_4 &= 0 \\ u_1 - u_2 - u_3 + u_4 &= 0 \end{aligned}$$

This system can be solved uniquely. The solution is

$$u_1 = u_2 = u_3 = u_4 = n/4$$

□

Corollary 1. *For any $n \times q$ matrix satisfying the conditions (i) and (ii) described in the introduction, it holds*
(a)

$$\begin{gathered} 0 \le u_1 + \ldots + u_{k/4} \le n/4, \\ 0 \le u_{k/4+1} + \ldots + u_{2k/4} \le n/4, \\ 0 \le u_{2k/4+1} + \ldots + u_{3k/4} \le n/4, \\ 0 \le u_{3k/4+1} + \ldots + u_k \le n/4, \end{gathered}$$

where $k = 2^{q-1}$.
(b) $0 \le u_i \le n/4, \quad i = 1, \ldots, 2^{q-1}$

Proof. We use the Distribution Lemma ignoring the last q-3 columns of the possible rows described in section 2. □

3.1 Parameters for q=3

The solution in paragraph 2.1 has one parameter. From Distribution Lemma it holds that $0 \le u_8 \le n/4$. Thus for each possible value of the parameter u_8 we can construct $n/4 + 1$ different $n \times 4$ matrices and by removing the first column have the corresponding $n/4 + 1$, not necessarily non isomorphic, OA(n,3,2,t).

3.2 Parameters for q=4

The solution in paragraph 2.2 has five parameters. From Distribution Lemma it holds that $0 \le u_1 + u_2 + u_3 + u_4 \le n/4$, $0 \le u_5 + u_6 + u_7 + u_8 \le n/4$, $0 \le u_9 + u_{10} + u_{11} + u_{12} \le n/4$, $0 \le u_{13} + u_{14} + u_{15} + u_{16} \le n/4$. Then for the parameters of the solution we have $0 \le u_8 \le n/4$, $0 \le u_{12} \le n/4$, $0 \le u_{14} + u_{15} + u_{16} \le n/4$. The total number of the possible values of the parameters is calculated to be

$$(n/4+1)^2 \begin{bmatrix} 4 \\ n/4 \end{bmatrix} = (n/4+1)^2 \binom{n/4+3}{n/4} = \frac{1}{6}(n/4+1)^3(n/4+2)(n/4+3)$$

where with $\begin{bmatrix} n \\ k \end{bmatrix} = \binom{n+k-1}{k}$ we denote the combinations with repetitions.

3.3 Parameters for q=5

For the parameters of the solution in paragraph 2.3, as previously, using the Distribution Lemma we can write the following restrictions $0 \leq u_8 \leq n/4$, $0 \leq u_{12} + u_{14} + u_{15} + u_{16} \leq n/4$, $0 \leq u_{20} + u_{22} + u_{23} + u_{24} \leq n/4$, $0 \leq u_{25} + \ldots + u_{31} \leq n/4$. The total number of the possible values of the parameters is calculated to be

$$(n/4+1)\begin{bmatrix}5\\n/4\end{bmatrix}\begin{bmatrix}5\\n/4\end{bmatrix}\begin{bmatrix}8\\n/4\end{bmatrix}=$$

$$\frac{1}{8!(4!)^2}(n/4+1)^4(n/4+2)^3(n/4+3)^3(n/4+4)^3(n/4+5)\ldots(n/4+8).$$

3.4 Parameters for q=6

For the parameters of the solution in paragraph 2.4, as previously, using the Distribution Lemma we can write the following restrictions $0 \leq u_8 + u_{12} + u_{14} + u_{15} + u_{16} \leq n/4$, $0 \leq u_{20} + u_{22} + u_{23} + u_{24} + u_{26} + \ldots + u_{32} \leq n/4$, $0 \leq u_{36} + u_{38} + u_{39} + u_{40} + u_{42} + \ldots + u_{48} \leq n/4$, $0 \leq u_{50} + \ldots + u_{64} \leq n/4$. The total number of the possible values of the parameters is calculated to be

$$\begin{bmatrix}6\\n/4\end{bmatrix}\begin{bmatrix}12\\n/4\end{bmatrix}\begin{bmatrix}12\\n/4\end{bmatrix}\begin{bmatrix}15\\n/4\end{bmatrix}=$$

$$\frac{(n/4+1)^4\ldots(n/4+5)^4(n/4+6)^3\ldots(n/4+11)^3(n/4+12)\ldots(n/4+14)}{5!(11!)^2 14!}.$$

3.5 Results

The number of OA(n,q,2,t) created using this method, for $q = 3, 4, 5$ and 6 and various n can been seen in table 1. We should note that the number of the arrays for $q > 3$ are less than the possible values of the parameters since from Distribution Lemma all u_i's must take values between zero and $n/4$. Many of these arrays are also isomorphic.

Table 1. Number of orthogonal arrays constructed

n\q	3	4	5	6
12	4	16	224	2688
16	5	51	1932	81420
20	6	96	10752	1869504
24	7	196	55284	42068304
28	8	336	244512	
32	9	581	959209	
36	10	912	3384032	
40	11	1422	10920184	
44	12	2096	32538016	

4 Implementation

The possible rows for any q can be easily generated if we take the binary representation of the numbers 0 to $2^q - 1$. Then replace the 1's with -1's and 0's with 1's. For a given q, let U be the $2^q \times (q+1)$ matrix for which the ith row is equal to $\underline{u}_i$, as described in section 2, $i = 1, \ldots, 2^q$. Then the matrix A of the system can be generated using the following algorithm:

```
s:=0;
for i from 1 to q do
    for j from i+1 to q+1 do
         s:=s+1;
         for z from 1 to 2^q do
            A(s,z):=U(z,i)*U(z,j);
s:=s+1;
A(s,:)=(1,...,1); #this is the for the condition
                  sum of ui's is equal n
```

The vector b for which $Au = b$ where $u = [u_1, \ldots u_{2^q}]$ is $b = [0, \ldots, 0, n]$. All systems were solved using Maple software. We used the package *linalg* and its routine *linsolve*, which solves the system using LU decomposition with pivoting when necessarily. Since the number of possible values for the parameters tends to grow fast as n increases we used C to quickly generate the parameters and get the exact solutions.

References

1. J. B. Clark and A. M. Dean, Equivalence of fractional factorial designs, *Statistica Sinica*, 11 (2001), 537-547
2. C. J. Colbourn, J. H. Dinitz and D. R. Stinson, Applications of combinatorial designs to communications, cryptography, and networking, Surveys in Combinatorics, 1999 (Canterbury), 37-100, *London Math. Soc. Lecture Note Ser.*, 267, Cambridge Univ. Press, Cambridge, 1999.
3. K.-T. Fang and G. Ge, A sensitive algorithm for detecting the inequivalence of Hadamard matrices, *Mathematics of Computation*, 73 (2003), 843-851.
4. S. Georgiou and C. Koukouvinos, New visual cryptographic schemes derived from orthogonal and mixed orthogonal arrays, *J. Discrete Math. Sci. Cryptogr.*, 7 (2004), 291-306.
5. A. S. Hedayat, N. J. A. Sloane and J. Stufken, *Orthogonal Arrays: Theory and Applications*, Springer-Verlag, New York, 1999.
6. Y. Li, New factorial designs that cannot be embedded into Hadamard matrices, *Int. J. Math.*, 2 (2002), 527-534.
7. C.-X. Ma, K.-T. Fang and D. K. J. Lin, On the isomorphism of fractional factorial designs, *Journal of Complexity*, 17 (2001), 86-97.
8. D.R. Stinson, *Combinatorial Designs: Construction and Analysis*, Springer-Verlag, New York, 2004.
9. D.R. Stinson, *Cryptography: Theory and Practice.* Second edition. CRC Press Series on Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton, FL, 2002.

Counting Techniques Specifying the Existence of Submatrices in Weighing Matrices

C. Kravvaritis[1,⋆], M. Mitrouli[1], and Jennifer Seberry[2]

[1] Department of Mathematics, University of Athens,
Panepistemiopolis 15784, Athens, Greece
{mmitroul, ckrav}@math.uoa.gr
[2] Centre for Computer Security Research, SITACS,
University of Wollongong, Wollongong, NSW, 2522, Australia
jennie@uow.edu.au

Abstract. Two algorithmic techniques for specifying the existence of a $k \times k$ submatrix with elements $0, \pm 1$ in a skew and symmetric conference matrix of order n are described. This specification is achieved using an appropriate computer algebra system.

Key Words and Phrases: Gaussian elimination, growth, complete pivoting, weighing matrices, symbolic computation.

AMS Subject Classification: 65F05, 65G05, 20B20.

1 Introduction

A $(0, 1, -1)$ matrix $W = W(n, k)$ of order n satisfying $WW^T = kI_n$ is called a *weighing matrix of order n and weight k* or simply a *weighing matrix.* A $W(n, n)$, $n \equiv 0 \,(\mathrm{mod}\, 4)$, is called a Hadamard matrix of order n. A $W = W(n, k)$ for which $W^T = -W$, $n \equiv 0 \,(\mathrm{mod}\, 4)$, is called a *skew–weighing matrix.* A $W = W(n, n-1)$ satisfying $W^T = W$, $n \equiv 2 \,(\mathrm{mod}\, 4)$, is called a *symmetric conference matrix.* Conference matrices cannot exist unless $n-1$ is the sum of two squares: thus they cannot exist for orders $22, 34, 58, 70, 78, 94$. For more details and construction of weighing matrices the reader can consult the book of Geramita and Seberry [2]. Two matrices are said to be *Hadamard equivalent* or *H-equivalent* if one can be obtained from the other by a sequence of the operations: 1. Interchange any pairs of rows and/or columns; 2. Multiply any rows and/or columns through by -1. Two important properties of the weighing matrices, which follow directly from the definition, are: 1. Every row and column of a $W(n, k)$ contains exactly $n - k$ zeros 2. Every two distinct rows and columns of a $W(n, k)$ are orthogonal to each other, which means that their inner product is zero.

The usefulness and significance of studying properties of the weighing matrices lies in the fact that they have applications in several scientific areas. They are

⋆ This research was financially supported by IKY (National Foundation for Scholarships).

used in Coding Theory for producing error correcting codes with good properties regarding the minimum Hamming distance. They appear also in the Theory of Statistical Designs and in Cryptography. One of their most important applications is in Numerical Analysis, and in particular in the study of the problem of the growth factor [1], which appears in the technique of Gaussian Elimination (GE) for solving a system of the form $A \cdot \underline{x} = \underline{b}$, where $A = [a_{ij}] \in \mathbb{R}^{n\times n}$ is nonsingular. According to known theorems [5] the accuracy of the computed solution with GE, which means in fact the stability of GE, depends on the growth factor. So, is created the growth problem, which is actually the problem of determining the growth factor for various values of the order n.

Experiments that have been made in the past on the computer reveal that the weighing matrices have certain interesting properties regarding the structure of the pivots appearing after GE. We are interested in specifying the existence of submatrices with the maximum value of determinant, which are embedded inside a weighing matrix. Write $W(j)$ for the absolute value of the determinant of the $j \times j$ principal submatrix in the upper left corner of the matrix W. It can be proved that the magnitude of the pivots appearing after the application of GE operations on a CP (completely pivoted, no exchanges are needed during GE with complete pivoting) matrix W are given by

$$p_j = W(j)/W(j-1), \quad j = 1, 2, \ldots, n, \quad W(0) = 1. \tag{1}$$

It is obvious from the previous relationship that principal determinants (minors) of a matrix are strictly connected with the appearing pivots after GE, since the value of a pivot is the quotient of two minors. The purpose of this paper is to demonstrate two algorithmic techniques, which will prove the existence of specific submatrices embedded in every $W(n, n-1)$, for appropriate value of n. This will be done by showing that in every $W(n, n-1)$ the columns, which make up these matrices, can always be found. We have achieved our goal by applying the notion of symbolic manipulation on a Computer Algebra Package, such as Maple. By assigning all possible values to our variables we perform complete (exhaustive) searches for all the appearing cases. This is a technique that is used over and over in Cryptography to find impossibilities and possibilities.

In [3], the pivot structure of $W(n, n-1)$ was studied and the problem of specifying specific $k \times k$ $(0, 1, -1)$ matrices existing embedded in $W(n, n-1)$ was initially posed.

Lemma 1. *The possible absolute values of the determinants of all* $n \times n$ $(0, 1, -1)$ *matrices, where there is at most one zero in each row and column, is given in Table 1 for* $n = 2, 3, 4, 5$.

In [4] were proved the following lemmas 2,3 and 4:

Lemma 2. *Let W be a CP skew and symmetric matrix, of order $n \geq 6$ then if GE is performed on W the first two pivots are* 1 *and* 2.

Lemma 3. *Let W be a CP skew and symmetric conference matrix, of order $n \geq 12$ then if GE is performed on W the third pivot is* 2 *and the fourth pivot is* 3 *or* 4.

Table 1. Determinant Values for $n = 2, 3, 4, 5$

Order	*Maximum Determinant*	*Possible Determinant Values*
2×2	2	$0, 1, 2$
3×3	4	$0, 1, 2, 3, 4$
4×4	16	$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 16$
5×5	48	$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 36, 40, 48$

2 Existence of Specific $(0, +, -)$ Submatrices in $W(n, n-1)$

Notation. Throughout this paper the elements of a $(0, 1, -1)$ matrix will be denoted by $(0, +, -)$. Let $\underline{y}^T_{\beta+1}$ the vectors containing the binary representation of each integer $\beta + 2^{k-1}$ for $\beta = 0, \ldots, 2^{k-1} - 1$. Replace all zero entries of $\underline{y}^T_{\beta+1}$ by -1 and define the $k \times 1$ vectors $\underline{u}_j = \underline{y}_{2^{k-1}-j+1}$, $j = 1, \ldots, 2^{k-1}$. We write U_k for all the $k \times (n - 2k + 1)$ matrices, in which $\underline{u}_j$ occurs x_j times. So

$$U_k = \begin{array}{ccccc}
\overbrace{+\ldots+}^{x_1} & \overbrace{+\ldots+}^{x_2} & \ldots & \overbrace{+\ldots+}^{x_{2^{k-1}-1}} & \overbrace{+\ldots+}^{x_{2^{k-1}}} \\
+\ldots+ & +\ldots+ & \ldots & -\ldots- & -\ldots- \\
\cdot & \cdot & \ldots & \cdot & \cdot \\
\cdot & \cdot & \ldots & \cdot & \cdot \\
+\ldots+ & +\ldots+ & \ldots & +\ldots+ & -\ldots- \\
+\ldots+ & -\ldots- & \ldots & +\ldots+ & -\ldots-
\end{array} = \begin{array}{ccccc}
x_1 & x_2 & \ldots & x_{2^{k-1}-1} & x_{2^{k-1}} \\
+ & + & \ldots & + & + \\
+ & + & \ldots & - & - \\
\vdots & \vdots & & \vdots & \vdots \\
+ & + & \ldots & - & - \\
+ & - & \ldots & + & -
\end{array}$$

where $x_1 + x_2 + \ldots + x_{2^{k-1}} = n - 2k + 1$.

Example 1.
$$U_3 = \begin{array}{cccc}
x_1 & x_2 & x_3 & x_4 \\
1 & 1 & 1 & 1 \\
1 & 1 & - & - \\
1 & - & 1 & -
\end{array}, \quad U_4 = \begin{array}{cccccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & - & - & - & - \\
1 & 1 & - & - & 1 & 1 & - & - \\
1 & - & 1 & - & 1 & - & 1 & -
\end{array}$$

2.1 An Algorithm Specifying the Existence of $k \times k$ $(0, +, -)$ Submatrices in a $W(n, n-1)$

The following algorithm specifies the existence of a $k \times k$ submatrix A in a $W(n, n-1)$, given that the upper left $(k-1) \times (k-1)$ submatrix B of A always exists in a $W(n, n-1)$.

```
Algorithm Exist 1
Step 1
Read the $k \times k$ matrix $A$ and the $(k-1) \times (k-1)$ matrix $B$
Step 2
Create the matrix $Z$
```

$$Z=\left[\begin{array}{c|c|cccccc} & & 0 & + & + & \cdots & \cdots & + \\ & & y_{21} & 0 & y_{23} & \cdots & \cdots & y_{2k}\\ B & U_k & y_{31} & y_{32} & 0 & y_{34} & \cdots & y_{3k}\\ & & \vdots & \vdots & & \vdots & \ddots & \vdots \\ \cline{1-1} + \; z_2 \cdots z_{k-1} & & y_{k1} & y_{k2} & y_{k3} & \cdots & y_{k,k-1} & 0\end{array}\right], \text{ where } z_i,\ y_{ij}=\pm 1$$

```
**If** $B$ contains columns with 0
    they are excluded from the matrix $Z(:, n-k+1:n)$
*Step 3*
**If** $A$ has r 0's
    **Demand** that the r columns of $Z(:, n-k+1:n)$, in which the 0's are in
    the same position as in $A$, take the appropriate values $y_{ij}$:
        they are identical with the r columns of $A$ containing the 0's
*Step 4*
**Procedure Solve**
**For** all possible values of $z_i$,$i=2,\dots,k-1$
    **Form** the system of $1+\binom{k}{2}$ equations and $2^{k-1}$ variables which results
    from counting of columns and the inner products of every two distinct rows
    **Solve** the system for all $x_i$
    **Find** the minimum values for the $x_i$ which correspond to the columns of
    $A$, given that the number of columns appearing in $Z(:, 1:k-1)$ is $\geq 1$
    **Formulate** (if necessary) conditions and/or restrictions for the order $n$
    or for some $x_i$:
        the columns of $A$ appear (the corresponding $x_i$ are all $\geq 1$)
**End**{of Procedure Solve}
**Else**
    **Do** Procedure Solve
```

Complexity. Obviously, all the calculations of the algorithm are made in Step 4, where the system is solved. The system has $m = 1+\binom{k}{2}$ equations with $v = 2^{k-1}$ variables, so it can be represented by an $m \times v$ matrix, with $v \geq m$. The solution of such a system requires about $f = m^2(v - \frac{m}{3})$ flops, if we use, for instance, QR factorization. Since the system is formed for all possible values of z_i, $i = 2, \dots, k-1$, and z_i can be ± 1, we have $2^{k-1-2+1} = 2^{k-2}$ systems. Hence, we have totally $f \cdot 2^{k-2}$ flops.

Comments.

1. Clearly, any arbitrary $W(n, n-1)$ can be written always in the form of the matrix Z and z_i, y_{ij} can be ± 1.
2. In Procedure Solve the system of $1+\binom{k}{2}$ equations and 2^{k-1} variables which results from the counting of all columns and the inner products of every two distinct rows, is formed only once. For every combination of all

possible values z_i,$i = 2, \ldots, k-1$, only the $k-1$ equations that result from the inner product between the $k-$th and the previous $k-1$ rows need to be changed every time.

3. Obviously, the system has exactly one solution only for k=3, otherwise it has infinite solutions, which are described by $2^{k-1} - 1 - \binom{k}{2}$ parameters.
4. If in the expression for the solution x_i appears n, we find the minimum value of n, for which we have $x_i \geq 1$. Otherwise, we either establish that always $x_i \geq 1$, or we apply conditions on the appearing parameters so that $x_i \geq 1$ holds.
5. If A contains some columns with 0's in its $(k-1) \times (k-1)$ upper left part, which is actually B, we exclude these columns from the submatrix $Z(:, n-k+1:n)$, and give to the corresponding z_i and y_{ij} appropriate values so that they are identical with the columns of A containing the 0's. If A contains a 0 in the k-th column or row, which means outside the $(k-1)\times(k-1)$ upper left sunmatrix B, then the corresponding column remains in $Z(:, n-k+1:n)$ and the variable z_i and y_{ij} in this column take appropriate values so that this column is identical with the one in A containing the 0 outside of B. After these subcases of Step 3 are examined, the matrix Z takes the desired form and the system is set up.
6. By saying "a submatrix A always exists in a $W(n, n-1)$" we mean actually that there exist always the columns of A in $W(n, n-1)$. Then, after a sequence of H-equivalent operations, A can appear on the upper left $k \times k$ block of the $W(n, n-1)$.
7. The Computer Package gives the ability to select the parameters among the variables before solving the system. In this way we take advantage of the appearance of the first $k-1$ columns of Z by assuming that the respective parameters are ≥ 1.

Implementation of the Algorithm Exist 1

Next we demonstrate the application of the above described Algorithm for various values of k.

1. Existence of 3×3 Matrices (k=3)

We want to establish whether the matrix

$$B_1 = \begin{bmatrix} + & + & + \\ + & - & + \\ + & + & - \end{bmatrix}$$

always exists in a $W(n, n-1)$. First we note that the upper left 2×2 submatrix of B_1 $\begin{bmatrix} + & + \\ + & - \end{bmatrix}$ always occurs in any $W(n, n-1)$, due to the orthogonality of the first two rows.

1. We have $A = B_1,\ B = \begin{bmatrix} + & + \\ + & - \end{bmatrix}$

2. We create

$$Z = \left[\begin{array}{cc|cccc|ccc} & & x_1 & x_2 & x_3 & x_4 & & & \\ + & + & \overbrace{+} & \overbrace{+} & \overbrace{+} & \overbrace{+} & 0 & + & + \\ + & - & + & + & - & - & u & 0 & w \\ \hline + & z & + & - & + & - & x & y & 0 \end{array}\right]$$

where u, w, x, y and z are ± 1.

3. **Case 1,** $z = 1$

For z=1, the system is

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &= n - 5 \\ x_1 + x_2 - x_3 - x_4 &= -w \\ x_1 - x_2 + x_3 - x_4 &= -2 - y \\ x_1 - x_2 - x_3 + x_4 &= -ux \end{aligned} \tag{2}$$

The system has exactly one solution, as we have 4 equations with 4 unknowns. The solution is:

$$\begin{aligned} x_1 &= \tfrac{1}{4}(-y + n - ux - 7 - w) \\ x_2 &= \tfrac{1}{4}(y + n + ux - 3 - w) \\ x_3 &= \tfrac{1}{4}(-7 + w - y + n + ux) \\ x_4 &= \tfrac{1}{4}(y + n - ux - 3 + w) \end{aligned} \tag{3}$$

We need to specify whether $x_2 \geq 1$, since the other two columns of A, $[+,+,+]^T$ *and* $[+,-,+]^T$, are the first two columns of Z. The minimum value of x_2 is $\frac{n}{4} - \frac{6}{4}$. We have

$$x_2 \geq \frac{n}{4} - \frac{6}{4} \geq 1 \Leftrightarrow x_2 \geq 1 \ for \ n \geq 10$$

Hence, we have that B_1 exists in any $W(n, n-1)$ with $n \geq 10$.

For $z = -1$, the system differs in the third and fourth equation. After calculating the solution, in this case we need to specify whether $x_2,\ x_3 \geq 1$, since the columns of A $[+,+,+]^T$ is the first column of Z. Similarly, we get $x_2,\ x_3 \geq 1$ for $n \geq 10$. Consequently B_1 exists in any $W(n, n-1)$ with $n \geq 10$.

With a similar argument we can prove that $B_2 = \begin{bmatrix} + & + & + \\ + & - & 0 \\ + & + & - \end{bmatrix}$ exists in any $W(n, n-1)$ with $n \geq 10$.

Lemma 4. *The matrices B_1 or B_2 always exist in a $W(n, n-1)$ with $n \geq 10$.*

Remark 1. The maximum value of the 3×3 minor of a $W(n, n-1)$ is equal, according to the previous results, to the absolute value of determinant of B_1 and B_2, which is 4.

With a similar argument we can prove the following Lemma:

Lemma 5. *The matrices* $A_1 = \begin{bmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{bmatrix}$ *or* $A_2 = \begin{bmatrix} + & + & 0 & - \\ + & - & - & - \\ + & - & + & + \\ + & + & - & + \end{bmatrix}$ *always exist in a* $W(n, n-1)$ *with* $n \geq 10$.

Remark 2. The maximum values of the 4×4 minors of a $W(n, n-1)$ are equal, according to the previous results, to the absolute values of determinants of A_1 and A_2, which are 16 and 12 respectively.

2. Existence of 5×5 Matrices (k=5)

We want to establish whether the matrix $C_8 = \begin{bmatrix} + & + & 0 & - & + \\ + & - & - & - & - \\ + & - & + & + & + \\ + & + & - & + & - \\ + & + & + & - & - \end{bmatrix}$ always exists in a $W(n, n-1)$. First we note that the upper left 4×4 submatrix of C_8 is A_2, which was proved previously that always occurs in any $W(n, n-1)$.

1. We have $A = C_8,\ B = A_2$
2. We create

$$Z = \left[\begin{array}{cccc|} + & + & 0 & - \\ + & - & - & - \\ + & - & + & + \\ + & + & - & + \\ \hline + & z & + & w \end{array}\right. \quad U_5 \quad \left|\begin{array}{cccc} + & + & + & + \\ 0 & a & b & c \\ d & 0 & e & f \\ g & h & 0 & k \\ l & m & p & 0 \end{array}\right]$$

where
$U_5 =$

$$\left[\begin{array}{cccccccccccccccc} \overbrace{}^{x_1} & \overbrace{}^{x_2} & \overbrace{}^{x_3} & \overbrace{}^{x_4} & \overbrace{}^{x_5} & \overbrace{}^{x_6} & \overbrace{}^{x_7} & \overbrace{}^{x_8} & \overbrace{}^{x_9} & \overbrace{}^{x_{10}} & \overbrace{}^{x_{11}} & \overbrace{}^{x_{12}} & \overbrace{}^{x_{13}} & \overbrace{}^{x_{14}} & \overbrace{}^{x_{15}} & \overbrace{}^{x_{16}} \\ + & + & + & + & + & + & + & + & + & + & + & + & + & + & + & + \\ + & + & + & + & + & + & + & + & - & - & - & - & - & - & - & - \\ + & + & + & + & - & - & - & - & + & + & + & - & - & - & + & - \\ + & + & - & - & + & + & - & - & + & + & - & + & - & + & - & - \\ + & - & + & - & + & - & + & - & + & - & + & + & + & - & - & - \end{array}\right]$$

and a, b, c, d, e, f, g, h, k, l, m, n, p, z and w, are ± 1. As described in Comment 5, the column $[0, -, +, -, +]^T$ is excluded from $Z(:, n-4:n)$ and the values of the variables in this column remain fixed.

3. **Case 1,** $z = 1$**,** $w = 1$
The system is

$$\begin{aligned}
&x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9+x_{10}+x_{11}+x_{12}+x_{13}+x_{14}\\
&+x_{15}+x_{16}=n-8\\
&x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8-x_9-x_{10}-x_{11}-x_{12}-x_{13}-x_{14}\\
&-x_{15}-x_{16}=-1-a-b-c\\
&x_1+x_2+x_3+x_4-x_5-x_6-x_7-x_8+x_9+x_{10}+x_{11}-x_{12}-x_{13}-x_{14}\\
&+x_{15}-x_{16}=1-d-e-f\\
&x_1+x_2-x_3-x_4+x_5+x_6-x_7-x_8+x_9+x_{10}-x_{11}+x_{12}-x_{13}+x_{14}\\
&-x_{15}-x_{16}=-1-g-h-k\\
&x_1-x_2+x_3-x_4+x_5-x_6+x_7-x_8+x_9-x_{10}+x_{11}+x_{12}+x_{13}-x_{14}\\
&-x_{15}-x_{16}=-1-l-m-p\\
&x_1+x_2+x_3+x_4-x_5-x_6-x_7-x_8-x_9-x_{10}-x_{11}+x_{12}+x_{13}+x_{14}\\
&-x_{15}+x_{16}=-be-cf\\
&x_1+x_2-x_3-x_4+x_5+x_6-x_7-x_8-x_9-x_{10}+x_{11}-x_{12}+x_{13}-x_{14}\\
&+x_{15}+x_{16}=-ah-ck\\
&x_1-x_2+x_3-x_4+x_5-x_6+x_7-x_8-x_9+x_{10}-x_{11}-x_{12}-x_{13}+x_{14}\\
&+x_{15}+x_{16}=2-am-bp\\
&x_1+x_2-x_3-x_4-x_5-x_6+x_7+x_8+x_9+x_{10}-x_{11}-x_{12}+x_{13}-x_{14}\\
&-x_{15}+x_{16}=-dg-fk\\
&x_1-x_2+x_3-x_4-x_5+x_6-x_7+x_8+x_9-x_{10}+x_{11}-x_{12}-x_{13}+x_{14}\\
&-x_{15}+x_{16}=-2-dl-ep\\
&x_1-x_2-x_3+x_4+x_5-x_6-x_7+x_8+x_9-x_{10}-x_{11}+x_{12}-x_{13}-x_{14}\\
&+x_{15}+x_{16}=-2-gl-hm
\end{aligned} \tag{4}$$

The system apparently has an infinite number of solutions, which depend on five parameters, as we have 11 equations with 16 unknowns. We have chosen between the five parameters x_1, x_8 and x_{12} because, in this case, the respective columns appear in Z and we want to make use of this fact by assuming $x_1,\ x_8,\ x_{12} \geq 1$. The other two parameters can be chosen arbitrary. The solution is:

$$\begin{aligned}
x_2 &= \tfrac{1}{4}(-f-a-b-fk-dg-ck-ah-c-d-e)-x_1+x_{12}+x_{14}\\
x_3 &= \tfrac{1}{4}(-8-be-cf-ep-dl-bp-am+n)-x_1-x_{14}-x_{16}\\
x_4 &= \tfrac{1}{4}(ep+dl+fk+dg+bp+am+ck+ah)+x_1-x_{12}+x_{16}\\
x_5 &= \tfrac{1}{4}(-g+ep+dl+fk+dg-l-h-k-m-p)+x_8-x_{12}+x_{16}\\
x_6 &= \tfrac{1}{4}(-10+f-ep-dl+l+n+d+e+m+p)-x_8-x_{14}-x_{16}\\
x_7 &= \tfrac{1}{4}(g-a-b+be+cf-fk-dg-c+h+k)-x_8+x_{12}+x_{14}\\
x_9 &= \tfrac{1}{4}(-12-ep-dl-fk-dg-hm-gl+n)-x_1-x_8-x_{16}\\
x_{10} &= \tfrac{1}{4}(4-g+a+b+ep+dl+fk+dg+ck+ah+hm+gl+c-h-k)\\
&\quad +x_1+x_8-x_{12}-x_{14}+x_{16}\\
x_{11} &= \tfrac{1}{4}(10-f+be+cf+ep+dl+fk+dg+bp+am+hm+gl-l-n\\
&\quad -d-e-m-p)+x_1+x_8+x_{14}+2x_{16}\\
x_{13} &= \tfrac{1}{4}(-8+f+a+b-be-cf+n+d+c+e)-x_{12}-x_{14}-x_{16}\\
x_{15} &= \tfrac{1}{4}(-8+g-ep-dl-fk-dg-bp-am-ck-ah-hm-gl+l+n\\
&\quad +m+h+k+p)-x_1-x_8+x_{12}-2x_{16}
\end{aligned} \tag{5}$$

We need to specify whether $x_7 \geq 1$, since the other columns of A appear in this case in Z. The minimum value of x_7 is $-\frac{10}{4} - x_8 + x_{12} + x_{14}$. We have

$$x_7 \geq -\frac{6}{4} - x_8 + x_{14} \geq 1 \Leftrightarrow x_{14} \geq 1 + \frac{6}{4} + x_8 \geq \frac{14}{4}$$

which means actually $x_{14} \geq 4$.

Hence, we have that C_8 exists in any $W(n, n-1)$ only if there exist at least 4 columns of the form $[+, -, -, +, -]^T$ or H-equivalent to it.
With similar arguments we deal with the other 3 cases and in every case it is proved that C_8 exists in any $W(n, n-1)$ if and only if $x_{14} \geq 4$.

Remark 3. It is obvious that for larger orders k the previous algorithm will encounter difficulties at extracting the wished results. Apart from this, results of the type "C_8 exists in any $W(n, n-1)$ if and only if $x_{14} \geq 4$" are not very general and consequently of less importance. So, we needed a more sophisticated technique which is more efficient in practice, provides more general results and can be used more easily for larger dimensions n.

2.2 Another Algorithm Specifying the Existence of $k \times k$ $(0, +, -)$ Submatrices in a $W(n, n-1)$

Notation. We denote by $U_{k,3}$ the first three rows of the previously defined matrix U_k.

$$U_{k,3} = \begin{array}{ccccc} x_1 & x_2 & \ldots & x_{2^{k-1}-1} & x_{2^{k-1}} \\ 1 & 1 & \ldots & 1 & 1 \\ 1 & 1 & \ldots & - & - \\ 1 & 1 & \ldots & - & - \end{array}$$

Example 2. $U_{3,3} = \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & - & - \\ 1 & - & 1 & - \end{array} = U_3$, $U_{4,3} = \begin{array}{cccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & - & - & - & - \\ 1 & 1 & - & - & 1 & 1 & - & - \end{array}$

The following algorithm specifies the existence of a $k \times k$ submatrix A in a $W(n, n-1)$, given that the upper left $(k-1) \times (k-1)$ submatrix B of A always exists in a $W(n, n-1)$.

Algorithm Exist 2
Step 1
Read the $k \times k$ matrix A and the $(k-1) \times (k-1)$ matrix B
Step 2
Denote with C the upper left $3 \times (k-1)$ submatrix of B
Step 3
Create the matrix Y_k

$$Y_k = \left[\begin{array}{c|c|cccccc} & & 0 & + & + & + & \cdots & + \\ C & U_{k,3} & y_{21} & 0 & y_{23} & y_{24} & \cdots & y_{2k} \\ & & y_{31} & y_{32} & 0 & y_{34} & \cdots & y_{3k} \end{array}\right], \text{ where } y_{ij} = \pm 1$$

Step 4
Compute the linear system resulting from the dimension and the orthogonality of the rows of Y_k and deduce the distribution of the number of columns of Y_k
Step 5
Find the maximum values for the x_i which correspond to the columns of A
Step 6
Formulate (if necessary) conditions for the order n:
the columns of A appear (the corresponding x_i are all ≥ 1)

Complexity. All the calculations of the algorithm are elementary and take place in Step 4. Precisely, we have totally $4 \cdot (2^{k-1} + 1)$ additions or subtractions, so the complexity is kept at low levels.

Comments.

1. Clearly, the first three rows of any arbitrary $W(n, n-1)$ can be written always in the form of the matrix Y_k and z_i, y_{ij} can be ± 1.
2. The matrix Y_k, which is created in Step 3, is in fact a submatrix of the matrix Z, as defined in the previous algorithm. In order to take advantage of the fact that B always exists, we include separately the first three rows of B in the matrix C.
3. In Step 4 is formulated a Distribution type Lemma, which gives the number of several columns appearing in a weighing matrix and will allow us to obtain bounds on the column structure of a weighing matrix. This Lemma results from the solution of the system, which is set up from counting of all columns and the inner products of every two distinct rows that must be zero.

Implementation of the Algorithm Exist 2

1. Existence of 5×5 Matrices (k=5)

We want to establish whether the matrix $C_1 = \begin{bmatrix} +&+&+&+&+ \\ +&-&+&-&- \\ +&-&-&+&+ \\ +&+&-&-&+ \\ +&+&-&+&- \end{bmatrix} = [x_1\ x_{12}\ x_8\ x_{11}\ x_{10}]$

always exists in a $W(n, n-1)$. First we note that the upper left 4×4 submatrix of C_1 is A_1, which was proved previously that always occurs in any $W(n, n-1)$.

1. We have $A = C_1,\ B = A_1$
2. $C = \begin{bmatrix} +&+&+&+ \\ +&-&+&- \\ +&-&-&+ \end{bmatrix}$

3. We create
$$Y_5 = \left[\begin{array}{cccc|c|ccccc} + & + & + & + & & 0 & + & + & + & + \\ + & - & + & - & U_{5,3} & q & 0 & a & b & c \\ + & - & - & + & & r & d & 0 & e & f \end{array}\right]$$
where a, b, c, d, e, f, q and r are ±1 and
$U_{5,3} =$
$$\left[\begin{array}{cccccccccccccccc} \overbrace{}^{x_1} & \overbrace{}^{x_2} & \overbrace{}^{x_3} & \overbrace{}^{x_4} & \overbrace{}^{x_5} & \overbrace{}^{x_6} & \overbrace{}^{x_7} & \overbrace{}^{x_8} & \overbrace{}^{x_9} & \overbrace{}^{x_{10}} & \overbrace{}^{x_{11}} & \overbrace{}^{x_{12}} & \overbrace{}^{x_{13}} & \overbrace{}^{x_{14}} & \overbrace{}^{x_{15}} & \overbrace{}^{x_{16}} \\ + & + & + & + & + & + & + & + & + & + & + & + & + & + & + & + \\ + & + & + & + & + & + & + & + & - & - & - & - & - & - & - & - \\ + & + & + & + & - & - & - & - & + & + & + & - & - & - & + & - \end{array}\right]$$

4. The Distribution Lemma for this case results from the following manipulation of the equations (they result from the dimension and the orthogonality):
$$x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9+x_{10}+x_{11}+x_{12}+x_{13}+x_{14}+x_{15}+x_{16} = n-9 \quad (6)$$
$$x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8-x_9-x_{10}-x_{11}-x_{12}-x_{13}-x_{14}-x_{15}-x_{16} = -a-b-c \quad (7)$$
$$x_1+x_2+x_3+x_4-x_5-x_6-x_7-x_8+x_9+x_{10}+x_{11}-x_{12}-x_{13}-x_{14}+x_{15}-x_{16} = -d-e-f \quad (8)$$
$$x_1+x_2+x_3+x_4-x_5-x_6-x_7-x_8-x_9-x_{10}-x_{11}+x_{12}+x_{13}+x_{14}-x_{15}+x_{16} = -qr-be-cf \quad (9)$$
$(6)+(7)+(8)+(9) : x_1+x_2+x_3+x_4 = \frac{1}{4}(n-9-a-b-c-g-h-k-qr-be-cf)$
$(6)+(7)-(8)-(9) : x_5+x_6+x_7+x_8 = \frac{1}{4}(n-9-a-b-c+d+e+f+qr+be+cf)$
$(6)-(7)+(8)-(9) : x_9+x_{10}+x_{11}+x_{12} = \frac{1}{4}(n-9+a+b+c-d-e-f+qr+be+cf)$
$(6)-(7)-(8)+(9) : x_{13}+x_{14}+x_{15}+x_{16} = \frac{1}{4}(n-9+a+b+c+d+e+f-qr-be-cf)$

Lemma 6. ***(Distribution Lemma)***. *Let W be any $W(n, n-1)$ of order $n > 2$ with its first three rows written in the form of Y_5. Then the number of columns which are*

(a) $(+,+,+)^T$ *or* $(-,-,-)^T$ *is* $\frac{1}{4}(n-9-a-b-c-g-h-k-qr-be-cf)$
(b) $(+,+,-)^T$ *or* $(-,-,+)^T$ *is* $\frac{1}{4}(n-9-a-b-c+d+e+f+qr+be+cf)$
(c) $(+,-,+)^T$ *or* $(-,+,-)^T$ *is* $\frac{1}{4}(n-9+a+b+c-d-e-f+qr+be+cf)$
(d) $(+,-,-)^T$ *or* $(-,+,+)^T$ *is* $\frac{1}{4}(n-9+a+b+c+d+e+f-qr-be-cf)$

5. We have obviously
$$x_1 \le \tfrac{1}{4}(n-9-a-b-c-g-h-k-qr-be-cf)$$
$$x_{10}, x_{11}, x_{12} \le \tfrac{1}{4}(n-9+a+b+c-d-e-f+qr+be+cf)$$
$$x_8 \le \tfrac{1}{4}(n-9-a-b-c+d+e+f+qr+be+cf)$$
By assigning all possible values ±1 to the variables, we get: $x_1, x_8, x_{10}, x_{11}, x_{12} \le \frac{1}{4}(n-4)$.
6. Hence, $1 \le x_1 \le \frac{1}{4}(n-4) \Leftrightarrow n \ge 8$. Similarly, $x_8, x_{10}, x_{11}, x_{12} \ge 1 \Leftrightarrow n \ge 8$. So we have that for $n \ge 8$ C_1 always exists. In a similar way can be proved the same result for the matrices in the following lemma.

Lemma 7. *One of the following matrices, named C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9 and C_{10} respectively,*

$$\begin{bmatrix} + & + & + & + & + \\ + & - & + & - & - \\ + & - & - & + & + \\ + & + & - & - & + \\ + & + & - & + & - \end{bmatrix} \begin{bmatrix} + & + & + & + & + \\ + & - & + & - & - \\ + & - & - & + & 0 \\ + & + & - & - & + \\ + & + & - & + & - \end{bmatrix} \begin{bmatrix} + & + & + & + & + \\ + & - & + & - & - \\ + & - & - & + & + \\ + & + & - & - & + \\ + & + & - & 0 & - \end{bmatrix} \begin{bmatrix} + & + & + & + & + \\ + & - & + & - & - \\ + & - & - & + & - \\ + & + & - & - & - \\ + & + & + & + & - \end{bmatrix} \begin{bmatrix} + & + & + & + & + \\ + & - & + & - & - \\ + & - & - & + & - \\ + & + & - & - & 0 \\ + & + & - & + & - \end{bmatrix}$$

$$\begin{bmatrix} + & + & + & + & + \\ + & - & + & - & - \\ + & - & - & + & - \\ + & + & - & - & + \\ + & + & 0 & - & - \end{bmatrix} \begin{bmatrix} + & + & + & + & + \\ + & - & + & - & - \\ + & - & - & + & + \\ + & + & - & - & + \\ + & + & - & + & 0 \end{bmatrix} \begin{bmatrix} + & + & 0 & - & + \\ + & - & - & - & - \\ + & - & + & + & + \\ + & + & - & + & - \\ + & + & + & - & - \end{bmatrix} \begin{bmatrix} + & + & 0 & - & + \\ + & - & - & - & - \\ + & - & + & + & 0 \\ + & + & - & + & - \\ + & + & + & - & - \end{bmatrix} \begin{bmatrix} + & + & 0 & - & + \\ + & - & - & - & - \\ + & - & + & + & + \\ + & + & - & + & 0 \\ + & + & + & 0 & - \end{bmatrix}$$

always exists in a $W(n, n-1)$ with $n \geq 8$.

Remark 4. The maximum values of the 5×5 minors of a $W(n, n-1)$ are equal, according to the previous results, to the absolute values of determinants of $C_1, \ldots, C_{10}$, which are 48, 40, 36 and 32.

Theorem 1. *Let W be a CP skew and symmetric conference matrix, of order $n \geq 8$ then if GE is performed on W the fifth pivot is 2 or 3 or $\frac{9}{4}$ or $\frac{10}{3}$ or $\frac{10}{4}$.*

Proof. It follows obviously from lemma 7, remark 4 and relationship (1). □

2.3 Conclusions

The object of our work was to find an algorithm able to decide if specific $(0, +, -)$ submatrices of order k exist embedded inside a CP weighing matrix $W(n, n-1)$. If such a submatrix exists, then, after a sequence of H-equivalent operations, it can appear on the upper left $k \times k$ block of the $W(n, n-1)$. So, the $k \times k$ principal minor of the $W(n, n-1)$ is equal to the determinant of this submatrix. Hence, according to relationship (1), the pivots of the $W(n, n-1)$ can be calculated.

By applying algorithms Exist 1 and Exist 2, we obtained the values for the fifth pivot of a CP skew and symmetric conference matrix of order $n \geq 8$. The use of symbolic algebra packages is required for the solution of the systems appearing in the implementation of the algorithms.

An issue open for research is to apply Algorithm Exist 2, or a more improved form of it, for the next orders of submatrices $k = 6, 7, \ldots$, in order to prove more appearing values in the pivot structure of a $W(n, n-1)$ for large enough n. Also the alteration of the parameters of the Algorithms, so that they can be used more generally for $W(n, n-p)$, $p = 2, 3, \ldots$, is interesting and is dealt with currently.

References

1. J. Day, and B. Peterson, Growth in Gaussian elimination, Amer. Math. Monthly, **95** (1988), 489-513.
2. A.V.Geramita, and J.Seberry, Orthogonal Designs: Quadratic Forms and Hadamard Matrices, Marcel Dekker, New York-Basel (1979).
3. C. Koukouvinos, M. Mitrouli and J. Seberry, Growth in Gaussian elimination for weighing matrices, $W(n, n-1)$, Linear Algebra Appl., **306** (2000), 189-202.
4. C. Kravvaritis, M. Mitrouli and J. Seberry, On the growth problem for skew and symmetric conference matrices, Linear Algebra Appl., **403** (2005), 183-296.
5. J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, London (1988).

Fast Verification for Respective Eigenvalues of Symmetric Matrix

Shinya Miyajima[1], Takeshi Ogita[1,2], and Shin'ichi Oishi[1]

[1] Faculty of Science and Engineering, Waseda University,
3-4-1 Okubo Shinjuku-ku Tokyo 169-8555 Japan
[2] CREST, Japan Science and Technology Agency
`shinya_miyajima@aoni.waseda.jp`

Abstract. A fast verification algorithm of calculating guaranteed error bounds for all approximate eigenvalues of a real symmetric matrix is proposed. In the proposed algorithm, Rump's and Wilkinson's bounds are combined. By introducing Wilkinson's bound, it is possible to improve the error bound obtained by the verification algorithm based on Rump's bound with a small additional cost. Finally, this paper includes some numerical examples to show the efficiency of the proposed algorithm.

1 Introduction

In this paper, we are concerned with the accuracy of computed eigenvalues for

$$Ax = \lambda x, \tag{1}$$

where A is a real symmetric $n \times n$ matrix, λ is an eigenvalue of A and x is an eigenvector corresponding to λ. There are several methods for calculating guaranteed error bounds for approximate eigenvalues and eigenvectors, e.g. [5,6,8,14]. Among them, we consider the case of verifying *all* eigenvalues in particular. With respect to the way of verifying a few *specified* eigenvalues, for example, see Yamamoto [14] and Golub, van Loan [2, p. 402, Theorem 8.1.16]. On error bounds for eigenvector, see Yamamoto [14] and Rump and Zemke [10]. Excellent overviews on perturbation theory for matrix eigenvalues can be found in Parlett [7], Stewart and Sun [11] and Ipsen [4].

The algorithms by Oishi [6] and Miyajima et al. [5] are known as the algorithm which calculates error bounds for all eigenvalues. Oishi's algorithm uses switches of rounding modes defined in IEEE standard 754 [1], especially rounding-upward and rounding-downward. In contrast, Miyajima's algorithm uses only rounding-to-nearest mode which is set in default, so that their algorithm does not require to change rounding modes.

In this paper, we introduce Rump's bound [9], which can supply *one* guaranteed error bound for all eigenvalues. However, this estimation may be pessimistic for relatively small eigenvalues. To overcome this, we introduce Wilkinson's bound [13]. This supplies an upper bound of distance between an approximate eigenvalue and its nearest true eigenvalue. Note that this does not always give an error bound of ith approximate eigenvalue for ith true eigenvalue.

The purpose of this paper is to propose a fast algorithm of calculating error bounds for respective eigenvalues by utilizing Rump's and Wilkinson's bounds. The proposed algorithm allows the presence of underflow in floating point arithmetic. After developing the verification algorithms named Rump's and Wilkinson's algorithms which are based on Rump's and Wilkinson's bounds, respectively, it is shown that the proposed algorithm supplies respective guaranteed error bounds where each of them is less than or equal to that by Rump's algorithm. Moreover, an intermediate result in the process of Rump's algorithm can be reused for Wilkinson's algorithm, so that additional computational cost against Rump's algorithm is almost negligible.

Finally, this paper includes some numerical examples to show the efficiency of the proposed algorithm.

2 Utilized Bounds

In this section, we briefly explain Rump's and Wilkinson's bounds and their properties. Hereafter, we assume that approximate eigenvalues $\tilde{\lambda}_i$ and eigenvevtors $\tilde{x}^{(i)}$ for $i = 1, \dots, n$ have already been computed.

Let $\tilde{D}$ and $\tilde{X}$ be an $n \times n$ diagonal matrix and an $n \times n$ approximately orthogonal matrix defined as $\tilde{D} := \mathrm{diag}(\tilde{\lambda}_1, \dots, \tilde{\lambda}_n)$ and $\tilde{X} := (\tilde{x}^{(1)}, \dots, \tilde{x}^{(n)})$, respectively. Then it is expected that $A\tilde{X} \approx \tilde{X}\tilde{D}$. We define $n \times n$ residual matrices R and G as

$$R := A\tilde{X} - \tilde{X}\tilde{D} \quad \text{and} \quad G := I - \tilde{X}^T\tilde{X}, \tag{2}$$

where I denotes the $n \times n$ identity matrix.

At first, we present Theorem 1 with respect to Rump's bound.

Theorem 1 (Rump [9]). *Let A be a real symmetric $n \times n$ matrix. Let λ_i and $\tilde{\lambda}_i$ for $i = 1, \dots, n$ be the true and approximate eigenvalues of A such that*

$$\lambda_1 \le \cdots \le \lambda_n \quad \textit{and} \quad \tilde{\lambda}_1 \le \cdots \le \tilde{\lambda}_n,$$

respectively. Let also R and G be defined as in (2). Then, it holds for all i that

$$|\lambda_i - \tilde{\lambda}_i| \le \frac{\|R\|_2}{1 - \|G\|_2}. \tag{3}$$

Comparing to matrix 1-norm and ∞-norm, it is disadvantageous in computational cost to compute matrix 2-norm with guaranteed accuracy. For a square matrix P, it is known that $\|P\|_2 \le \sqrt{\|P\|_1\|P\|_\infty}$. Moreover, if P is symmetric in particular, then it follows that $\|P\|_2 \le \|P\|_\infty$. Thus, we obtain

$$\frac{\|R\|_2}{1 - \|G\|_2} \le \frac{\sqrt{\|R\|_1\|R\|_\infty}}{1 - \|G\|_\infty}. \tag{4}$$

In this paper, Rump's algorithm is implemented based on the following formula instead of (3):

$$|\lambda_i - \tilde{\lambda}_i| \le \frac{\sqrt{\|R\|_1\|R\|_\infty}}{1 - \|G\|_\infty}. \tag{5}$$

The right hand side of (5) does not depend on indices i. Therefore, Rump's algorithm can supply one error bound for all eigenvalues. However, the error bound may be pessimistic for relatively small eigenvalues. For example, let A be a 2×2 matrix and assume $\lambda_1 = 1$, $\lambda_2 = 100$ and $\tilde{\lambda}_1 = 0.9$, $\tilde{\lambda}_2 = 95$. Then the algorithm tells $|\lambda_i - \tilde{\lambda}_i| \leq 5$, whereas $|\lambda_1 - \tilde{\lambda}_1| = 0.1$, that is relatively 500% overestimated.

To improve this, we present Theorem 2 with respect to Wilkinson's bound.

Theorem 2 (Wilkinson [13]). *Let A, $\tilde{\lambda}_i$ and λ_j be defined as in Theorem 1. Let also $\tilde{x}^{(i)}$ be the ith eigenvector of A corresponding to $\tilde{\lambda}_i$. Then*

$$\min_{1 \leq j \leq n} |\lambda_j - \tilde{\lambda}_i| \leq \frac{\|r^{(i)}\|_2}{\|\tilde{x}^{(i)}\|_2}, \tag{6}$$

where

$$r^{(i)} := A\tilde{x}^{(i)} - \tilde{\lambda}_i \tilde{x}^{(i)}. \tag{7}$$

Theorem 2 supplies an upper bound of distance between $\tilde{\lambda}_i$ and λ_j which lies nearest from $\tilde{\lambda}_i$. Hence, it does not always follows that $\min_{1 \leq j \leq n} |\lambda_j - \tilde{\lambda}_i| = |\lambda_i - \tilde{\lambda}_i|$. For example, if the eigenvalues are closely clustered, it can occur that $\min_{1 \leq j \leq n} |\lambda_j - \tilde{\lambda}_i| \neq |\lambda_i - \tilde{\lambda}_i|$ i.e. $j \neq i$.

In the next section, we will explain how Theorems 1 and 2 are related and used.

3 Proposed Algorithm

In this section, we propose a fast algorithm of calculating the error bounds for respective eigenvalues by utilizing Theorems 1 and 2. At first, we present the ground for combining (5) and (6), and show that the proposed algorithm supplies error bounds where each of them is less than or equal to that by Rump's algorithm. Moreover, we propose the way to reduce the computational cost for computing the right hand side of (6) by reusing an intermediate result in the process of Rump's algorithm. Next, we propose the way to check whether Theorem 2 can be applied to calculate an upper bound of $|\lambda_i - \tilde{\lambda}_i|$ for each i. Finally in this section, we describe the concrete steps of the proposed algorithm.

We first present Lemmas 1 and 2 as the preparation to prove Theorem 3.

Lemma 1. *Let $q^{(j)}$ be the jth column of a real $n \times n$ matrix Q. Then it holds for $j = 1, \ldots, n$ that*

$$\|q^{(j)}\|_2 \leq \sqrt{\|Q\|_1 \|Q\|_\infty}. \tag{8}$$

Proof of Lemma 1. Let q_{ij} be the (i, j) element of Q. Since both sides of (8) are nonnegative, (8) is equivalent to

$$\sum_{i=1}^{n} q_{ij}^2 \leq \|Q\|_1 \|Q\|_\infty. \tag{9}$$

Therefore, we aim to prove (9) instead of (8). Considering the right hand side of (9) yields

$$\begin{aligned}\|Q\|_1\|Q\|_\infty &= \left(\max_{1\le j\le n}\sum_{i=1}^{n}|q_{ij}|\right)\left(\max_{1\le i\le n}\sum_{j=1}^{n}|q_{ij}|\right)\\ &\ge \max_{1\le j\le n}\sum_{i=1}^{n}|q_{ij}||q_{ij}| = \max_{1\le j\le n}\sum_{i=1}^{n}q_{ij}^2 \ge \sum_{i=1}^{n}q_{ij}^2, \qquad (10)\end{aligned}$$

which proves (9) and Lemma 1. □

Lemma 2. *Let $\tilde{Q}$ be a real $n \times n$ matrix and $\tilde{q}^{(j)}$ the jth column of $\tilde{Q}$, then it holds that*

$$1 - \|I - \tilde{Q}^T\tilde{Q}\|_\infty \le \|\tilde{q}^{(i)}\|_2, \qquad (11)$$

where I is the $n \times n$ identity matrix.

Proof of Lemma 2. In the case $\|\tilde{q}^{(i)}\|_2 \ge 1$, Lemma 2 is obvious. Therefore, we consider the case

$$\|\tilde{q}^{(i)}\|_2 < 1. \qquad (12)$$

For any real orthonormal $n \times n$ matrix Q, there exist real $n \times n$ matrices E and F such that

$$\tilde{Q} = Q + E \qquad (13)$$

$$\tilde{Q}^T\tilde{Q} = I + F. \qquad (14)$$

Let $q^{(i)}$ and $e^{(i)}$ be the ith column of Q and E, respectively. Then it follows for $i = 1, \ldots, n$ that

$$\tilde{q}^{(i)} = q^{(i)} + e^{(i)}. \qquad (15)$$

From (12), (14), (15) and $\|q^{(i)}\|_2 = 1$, it follows that

$$\begin{aligned}\|I - \tilde{Q}^T\tilde{Q}\|_\infty + \|\tilde{q}^{(i)}\|_2 &= \|F\|_\infty + \|q^{(i)} + e^{(i)}\|_2\\ &\ge \|F\|_\infty + \|q^{(i)} + e^{(i)}\|_2^2\\ &= \|F\|_\infty + q^{(i)T}q^{(i)} + 2q^{(i)T}e^{(i)} + e^{(i)T}e^{(i)}\\ &= 1 + \|F\|_\infty + 2q^{(i)T}e^{(i)} + e^{(i)T}e^{(i)}. \qquad (16)\end{aligned}$$

On the ohter hand, utilizing (13) and the orthogonality of Q yields

$$\begin{aligned}\tilde{Q}^T\tilde{Q} &= (Q + E)^T(Q + E)\\ &= I + Q^TE + E^TQ + E^TE. \qquad (17)\end{aligned}$$

From (14) and (17), we obtain

$$F = Q^TE + E^TQ + E^TE. \qquad (18)$$

We here define the $n \times n$ diagonal matrix F_d whose diagonal elements are equal to those of F as follows:

$$F_d := \begin{pmatrix} f_{11} & & O \\ & \ddots & \\ O & & f_{nn} \end{pmatrix}, \quad f_{ii} := 2q^{(i)T}e^{(i)} + e^{(i)T}e^{(i)}. \tag{19}$$

Then it is obvious that

$$\|F_d\|_\infty \le \|F\|_\infty. \tag{20}$$

Inserting (20) into (16) yields

$$\begin{aligned} \|I - \tilde{Q}^T\tilde{Q}\|_\infty + \|\tilde{q}^{(i)}\|_2 \ge 1 + \|F_d\|_\infty + f_{ii} &= 1 + \max_{1\le k\le n} |f_{kk}| + f_{ii} \\ &\ge 1 + |f_{ii}| + f_{ii} \ge 1, \end{aligned} \tag{21}$$

which proves Lemma 2. □

From Lemmas 1 and 2, we finally have Theorem 3.

Theorem 3. *Let $\tilde{x}^{(i)}$ be defined as in Theorem 2. Let also R, G and $r^{(i)}$ be defined as in (2) and (7). Then the following inequality holds:*

$$\frac{\|r^{(i)}\|_2}{\|\tilde{x}^{(i)}\|_2} \le \frac{\sqrt{\|R\|_1\|R\|_\infty}}{1 - \|G\|_\infty}. \tag{22}$$

Let δ and ε_i for $i = 1, \dots, n$ be defined by

$$\delta := \frac{\sqrt{\|R\|_1\|R\|_\infty}}{1 - \|G\|_\infty} \quad \text{and} \quad \varepsilon_i := \frac{\|r^{(i)}\|_2}{\|\tilde{x}^{(i)}\|_2}. \tag{23}$$

From Theorem 3, it follows that $\varepsilon_i \le \delta$ for $i = 1, \dots, n$. We will design the proposed algorithm to supply the error bounds η_i such that

$$\eta_i = \begin{cases} \varepsilon_i & (\text{if it is proved that } \min_{1\le j\le n} |\lambda_j - \tilde{\lambda}_i| = |\lambda_i - \tilde{\lambda}_i|) \\ \delta & (\text{otherwise}) \end{cases}. \tag{24}$$

Therefore, it is guaranteed that the proposed algorithm can give the error bounds such that $\eta_i \le \delta$ for all i.

To obtain ε_i, we need to compute $r^{(i)} = A\tilde{x}^{(i)} - \tilde{\lambda}_i\tilde{x}^{(i)}$. Here, $r^{(i)}$ is the identical to the ith column of R. So, if R has already been obtained in the process of calculating δ, we can reuse it for calculating ε_i, $i = 1, \dots, n$. By this reuse, computational cost of the proposed algorithm can significantly be reduced. Moreover, utilizing a priori error estimation used in [3,5] and considering the presence of underflow yields

$$\|I - \tilde{X}^T\tilde{X}\|_\infty \le \|\mathrm{fl}_\square(I - \tilde{X}^T\tilde{X})\|_\infty + \gamma(\||\tilde{X}^T|(|\tilde{X}|s)\|_\infty + 1) + n\underline{\mathbf{u}} \tag{25}$$

where $s := (1,\dots,1)^T$, γ is defined as $\gamma := \frac{(n+1)\mathbf{u}}{1-(n+1)\mathbf{u}}$ with $\mathbf{u}$ being the unit roundoff, and $\underline{\mathbf{u}}$ as underflow unit (especially, $\mathbf{u} = 2^{-53}$ and $\underline{\mathbf{u}} = 2^{-1074}$ in IEEE 754 double precision). Throughout this paper, $\mathrm{fl}(\cdot)$ denotes the result of floating point computations, where all operations inside parentheses are executed by ordinary floating point arithmetic fulfilling rounding mode instruction, especially $\mathrm{fl}_\square(\cdot)$ in rounding-to-nearest, $\mathrm{fl}_\triangle(\cdot)$ in rounding-upward and $\mathrm{fl}_\nabla(\cdot)$ in rounding-downward.

By this way, we can avoid to execute matrix multiplication twice (once in rounding-upward and once in rounding-downward) as done in [6] to obtain the rigorous enclosure of $\tilde{X}^T\tilde{X}$. We need to execute matrix multiplication only once in rounding-to-nearest. Thus computational cost for $\|\mathrm{fl}_\square(I - \tilde{X}^T\tilde{X})\|_\infty$ is n^3 flops. Moreover, computational cost for $\||\tilde{X}^T|(|\tilde{X}|s)\|_\infty$ is $O(n^2)$ flops.

Based on the above discussions, we present concrete steps of computations for δ and $\varepsilon_1,\dots,\varepsilon_n$ in Algorithms 1 and 2 assuming that $\tilde{D}$ and $\tilde{X}$ have already been obtained. We express the algorithms Matlab-like [12].

Algorithm 1. *Computation of rigorous upper bound δ for*

$$\frac{\sqrt{\|R\|_1\|R\|_\infty}}{1-\|G\|_\infty} \le \delta$$

utilizing $\tilde{D}$ and $\tilde{X}$ which have already been obtained. R is also outputted to reuse in Algorithm 2. Computational cost of Algorithm 1 is $5n^3$ flops.

```
function $[\delta, R] = \mathrm{veigR}(A, \tilde{D}, \tilde{X})$
$Y = \mathrm{fl}_\square(I - \tilde{X}^T\tilde{X})$;
$\overline{B} = \mathrm{fl}_\triangle(A\tilde{X})$;   $\overline{T} = \mathrm{fl}_\triangle(\tilde{X}\tilde{D})$;
$\underline{B} = \mathrm{fl}_\nabla(A\tilde{X})$;   $\underline{T} = \mathrm{fl}_\nabla(\tilde{X}\tilde{D})$;
$\underline{R} = \mathrm{fl}_\nabla(\underline{B} - \overline{T})$;
$\overline{R} = \mathrm{fl}_\triangle(\overline{B} - \underline{T})$;
$\overline{R} = \max(|\underline{R}|, |\overline{R}|)$;
$\gamma = \mathrm{fl}_\triangle\left((n+1)\mathbf{u}/(1-(n+1)\mathbf{u})\right)$;
$p = \mathrm{fl}_\triangle(\|Y\|_\infty + (\gamma(\||\tilde{X}^T|(|\tilde{X}|s)\|_\infty + 1) + n\underline{\mathbf{u}}))$;   % $s = (1,\dots,1)^T$
if $p \ge 1$, error('Verification failed'), end
$q = \mathrm{fl}_\triangle(\|\overline{R}\|_1)$;
$r = \mathrm{fl}_\triangle(\|\overline{R}\|_\infty)$;
$\delta = \mathrm{fl}_\triangle\left(\sqrt{qr}/-(p-1)\right)$;   % $\mathrm{fl}_\triangle(-(p-1)) \le 1-p$
```

If $p \ge 1$ in Algorithm 1, it may be $\|G\|_\infty = \|I - \tilde{X}^T\tilde{X}\|_\infty \ge 1$ so that we cannot use (5). However, such a case seems to hardly occur because condition number on eigenvalue for a symmetric matrix is always one, i.e., $\|X\|_2\|X^T\|_2 = \|X^TX\|_2 = 1$. Therefore it only depends on numerical stability of calculating $\tilde{X}$.

Algorithm 2. *Computation of rigorous upper bound ε_i for*

$$\frac{\|r^{(i)}\|_2}{\|\tilde{x}^{(i)}\|_2} \le \varepsilon_i \quad \forall i = 1,\dots,n$$

utilizing R and $\tilde{X}$ which have already been obtained. Computational cost of Algorithm 2 is $4n^2$ flops.

```
function $[\varepsilon_1, \dots, \varepsilon_n] = \mathrm{veigW}(R, \tilde{X})$
for $i = 1 : n$
    $p_i = \mathrm{fl}_\bigtriangledown(\|\tilde{x}^{(i)}\|_2)$;    % $\tilde{x}^{(i)}$: the ith column of $\tilde{X}$ (approximate eigenvector)
end
for $i = 1 : n$
    $q_i = \mathrm{fl}_\bigtriangleup(\|r^{(i)}\|_2)$;    % $r^{(i)}$: the ith column of $R$
    $\varepsilon_i = \mathrm{fl}_\bigtriangleup(q_i/p_i)$;
end
```

Next, we present Theorems 4 and 5 to check whether $\min_{1\le j\le n} |\lambda_j - \tilde{\lambda}_i| = |\lambda_i - \tilde{\lambda}_i|$ holds.

Theorem 4. *Let λ_i and $\tilde{\lambda}_i$ be defined as in Theorem 1. Let also δ be defined as in (23). Suppose that*

$$\begin{cases} \tilde{\lambda}_{i+1} - \tilde{\lambda}_i > 2\delta & & (i = 1) \\ \tilde{\lambda}_i - \tilde{\lambda}_{i-1} > 2\delta \quad \wedge \quad \tilde{\lambda}_{i+1} - \tilde{\lambda}_i > 2\delta & & (2 \le i \le n-1) \\ \tilde{\lambda}_i - \tilde{\lambda}_{i-1} > 2\delta & & (i = n) \end{cases} \tag{26}$$

holds for some i. Then

$$\min_{1\le j\le n} |\lambda_j - \tilde{\lambda}_i| = |\lambda_i - \tilde{\lambda}_i|.$$

Proof of Theorem 4. From (5), it follows that $|\lambda_i - \tilde{\lambda}_i| \le \delta$. Therefore, λ_i exists in the interval $[\tilde{\lambda}_i - \delta, \tilde{\lambda}_i + \delta]$. Moreover, it is made sure from (26) that the eigenvalue which exists in $[\tilde{\lambda}_i - \delta, \tilde{\lambda}_i + \delta]$ is λ_i only (see Fig. 1). Hence, we can definitely say that the eigenvalue which lies nearest from $\tilde{\lambda}_i$ is λ_i, which proves Theorem 4. □

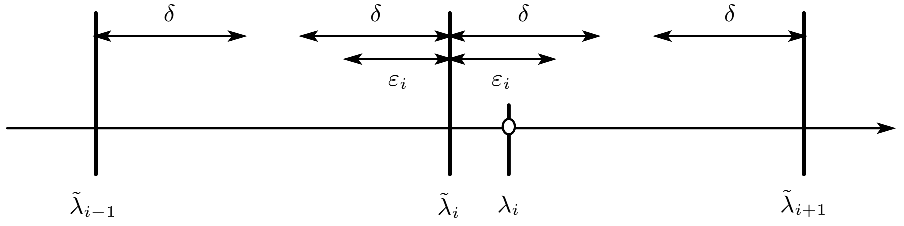


Fig. 1. The case that $\tilde{\lambda}_i - \tilde{\lambda}_{i-1} > 2\delta \ \wedge \ \tilde{\lambda}_{i+1} - \tilde{\lambda}_i > 2\delta$ holds

As we said above, condition number on eigenvalue for a symmetric matrix is always one. Only we have to consider is the case where eigenvalues are clustered. Therefore we next consider ill-conditioned case where some eigenvalues are nearly multiple. If the distance between them is less than δ, then (26) does not hold. As the fallback, we present Theorem 5 which supplies the signpost to check whether $\min_{1\le j\le n} |\lambda_j - \tilde{\lambda}_i| = |\lambda_i - \tilde{\lambda}_i|$ holds or not.

Theorem 5. *Let λ_i and $\tilde{\lambda}_i$ be defined as in Theorem 1. Let also δ and ε_i be defined as in (23) and (23) . Suppose that some approximate eigenvalues $\tilde{\lambda}_{\underline{k}}, \ldots, \tilde{\lambda}_{\overline{k}}$ with $1 \le \underline{k} < \overline{k} \le n$ are clustered such that*

$$\tilde{\lambda}_{\underline{k}} - \tilde{\lambda}_{\underline{k}-1} > 2\delta \ \wedge \ \tilde{\lambda}_{\overline{k}+1} - \tilde{\lambda}_{\overline{k}} > 2\delta \ \wedge \ \tilde{\lambda}_{i+1} - \tilde{\lambda}_i \le 2\delta \tag{27}$$

for all $i = \underline{k}, \ldots, \overline{k} - 1$. If it holds for all $i = \underline{k}, \ldots, \overline{k} - 1$ that

$$\varepsilon_i + \varepsilon_{i+1} < \tilde{\lambda}_{i+1} - \tilde{\lambda}_i, \tag{28}$$

then

$$\min_{1 \le j \le n} |\lambda_j - \tilde{\lambda}_i| = |\lambda_i - \tilde{\lambda}_i| \quad \textit{for all } i = \underline{k}, \ldots, \overline{k}.$$

Proof of Theorem 5. From Theorem 3, it holds for all $i = 1, \ldots, n$ that $\varepsilon_i \le \delta$. Moreover, from (27) and (28), the eigenvalue λ_i is the only one which exists in the interval $[\tilde{\lambda}_i - \varepsilon_i, \tilde{\lambda}_i + \varepsilon_i]$ for each i (see Fig. 2). Hence, we can definitely say that the eigenvalue which lies nearest from $\tilde{\lambda}_i$ is λ_i. This proves Theorem 5. □

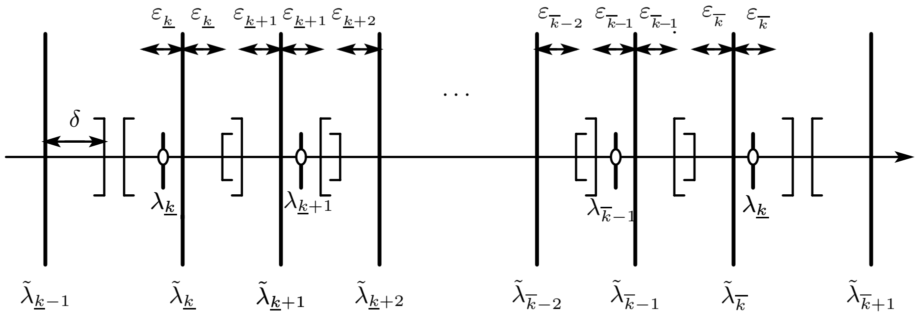

$\varepsilon_{\underline{k}}$ $\varepsilon_{\underline{k}}$ $\varepsilon_{\underline{k}+1}$ $\varepsilon_{\underline{k}+1}$ $\varepsilon_{\underline{k}+2}$
$\varepsilon_{\overline{k}-2}$ $\varepsilon_{\overline{k}-1}$ $\varepsilon_{\overline{k}-1}$ $\varepsilon_{\overline{k}}$ $\varepsilon_{\overline{k}}$
δ
$\lambda_{\underline{k}}$
$\lambda_{\underline{k}+1}$
$\lambda_{\overline{k}-1}$
$\lambda_{\overline{k}}$
$\tilde{\lambda}_{\underline{k}-1}$ $\tilde{\lambda}_{\underline{k}}$ $\tilde{\lambda}_{\underline{k}+1}$ $\tilde{\lambda}_{\underline{k}+2}$ $\tilde{\lambda}_{\overline{k}-2}$ $\tilde{\lambda}_{\overline{k}-1}$ $\tilde{\lambda}_{\overline{k}}$ $\tilde{\lambda}_{\overline{k}+1}$

Fig. 2. The case that (27) holds

Based on the above mentioned discussions, we present the concrete steps of the proposed algorithm utilizing Theorems 3, 4 and 5.

Algorithm 3. *Computation of error bounds η_i for respective eigenvalues $\tilde{\lambda}_i$ combining (5) and (6) on the assumption that all $\tilde{\lambda}_i$ and $\tilde{x}^{(i)}$ have already been obtained. Computational cost of Algorithm 3 is $5n^3$ flops.*

Step 1: *Compute δ by use of Algorithm 1.*
Step 2: *Compute $\varepsilon_1, \ldots, \varepsilon_n$ by applying Algorithm 2.*
Step 3: *For $i = 1, \ldots, n$, check whether*

$$\begin{cases} \mathrm{fl}_{\triangledown}(\tilde{\lambda}_{i+1} - \tilde{\lambda}_i) > 2\delta & & (i = 1) \\ \mathrm{fl}_{\triangledown}(\tilde{\lambda}_i - \tilde{\lambda}_{i-1}) > 2\delta \quad \wedge \quad \mathrm{fl}_{\triangledown}(\tilde{\lambda}_{i+1} - \tilde{\lambda}_i) > 2\delta & & (2 \le i \le n-1) \\ \mathrm{fl}_{\triangledown}(\tilde{\lambda}_i - \tilde{\lambda}_{i-1}) > 2\delta & & (i = n) \end{cases} \tag{29}$$

Table 1. η, δ and t corresponding to several algorithms

$\eta^{(1)},\ t_1$	$\eta^{(2)},\ t_2$	$\eta^{(3)},\ t_3$	$\delta,\ t_\delta$
Oishi [6, Alg. 3]	Miyajima et al. [5, Alg. 3]	Alg. 3	Alg. 1

holds or not. If yes, set the error bound η_i *as*

$$\eta_i = \varepsilon_i.$$

Otherwise, group the clustered eigenvalues as

$$\Lambda_1, \ldots, \Lambda_m \ (m < n), \quad \Lambda_j = \{\tilde{\lambda}_{\underline{k}_j}, \ldots, \tilde{\lambda}_{\overline{k}_j}\}.$$

Step4: *For* $j = 1, \ldots, m$ *with* $\Lambda_j = \{\tilde{\lambda}_{\underline{k}_j}, \ldots, \tilde{\lambda}_{\overline{k}_j}\}$, *check whether*

$$\mathrm{fl}_\triangle(\varepsilon_i + \varepsilon_{i+1}) < \mathrm{fl}_\bigtriangledown(\tilde{\lambda}_{i+1} - \tilde{\lambda}_i) \quad \forall i = \underline{k}_j, \ldots, \overline{k}_j - 1 \tag{30}$$

holds or not. If yes, set the error bound η_i *as*

$$\eta_i = \varepsilon_i \quad \forall i = \underline{k}_j, \ldots, \overline{k}_j.$$

Otherwise, set the error bound η_i *as*

$$\eta_i = \delta \quad \forall i = \underline{k}_j, \ldots, \overline{k}_j.$$

Computational cost of **Step 1** is $5n^3$ flops and that from **Step 2** to **Step 4** is only $O(n^2)$ flops, so that total computational cost of Algorithm 3 is still $5n^3$ flops, which is the same as that of Algorithm 1. Therefore, as n increases, computing time for **Step 1** becomes dominant and that from **Step 2** to **Step 4** does not mostly influence the total computing time.

4 Numerical Examples

In this section, we report some numerical results to show the efficiency of the proposed algorithm. Here, our computer environment is Pentium IV 3.4GHz CPU. We use Matlab 7.0 with ATLAS and IEEE 754 double precision for all computations.

Let η and t be an n-vector whose entries are guaranteed error bounds of eigenvalues and computing time (sec) resulted by the algorithms in Table 1. Let also t_{eig} and t_{vec} be the computing time (sec) for calculating all eigenvalues and eigenvectors, respectively.

At first, consider the following real symmetric 3×3 matrix A:

$$A := \mathrm{fl}((B + B^T)/2), \qquad B := \mathrm{fl}(XDX^T) \tag{31}$$

where $D := \mathrm{diag}(1, 1+\theta, 100), \theta := 2^{-52}$ and

$$X := \begin{pmatrix} -0.86584525931213 & -0.41783442087899 & 0.27518427218037 \\ -0.16528548023377 & 0.75803310394750 & 0.63092513291484 \\ -0.47222102552789 & 0.50079957073507 & -0.72540133236778 \end{pmatrix}.$$

Here, X is an approximately orthogonal matrix. Then the true eigenvalues of A should be near from

$$\lambda_1 = 1, \quad \lambda_2 = 1 + \theta, \quad \lambda_3 = 100.$$

The approximate eigenvalues $\tilde{\lambda}_i$ obtained in by Matlab function `eig` were

$$\tilde{\lambda}_1 = 0.99999999999997, \quad \tilde{\lambda}_2 = 1.00000000000000, \quad \tilde{\lambda}_3 = 99.9999999999997\,.$$

Then we obtained

$$\delta = 6.359\text{e-}14, \quad \varepsilon_1 = 3.385\text{e-}14, \quad \varepsilon_2 = 8.487\text{e-}15, \quad \varepsilon_3 = 4.159\text{e-}14$$

by Algorithms 1 and 2. In this case, the error bounds $\eta^{(3)}$ were

$$\eta_1^{(3)} = \delta = 6.359\text{e-}14, \quad \eta_2^{(3)} = \delta = 6.359\text{e-}14, \quad \eta_3^{(3)} = \varepsilon_3 = 4.159\text{e-}14$$

because neither (29) nor (30) holds for $\tilde{\lambda}_1$ and $\tilde{\lambda}_2$. In this connection, the error bounds $\eta^{(1)}$ and $\eta^{(2)}$ were

$$\eta_1^{(1)} = 6.837\text{e-}14, \quad \eta_2^{(1)} = 6.837\text{e-}14, \quad \eta_3^{(1)} = 1.536\text{e-}13$$

and

$$\eta_1^{(2)} = 1.789\text{e-}13, \quad \eta_2^{(2)} = 1.789\text{e-}13, \quad \eta_3^{(2)} = 4.266\text{e-}13.$$

This result shows that Algorithm 3 supplies a smaller error bound for $\tilde{\lambda}_3$ than δ in this example. We also confirm that Algorithms 1 and 3 supply smaller error bounds for all $\tilde{\lambda}_i$ than those by the algorithms proposed in [6] and [5].

Next, let A be a real symmetric $n \times n$ matrix whose entries are pseudo-random numbers uniformly distributed in $[-1, 1]$. Table 2 displays $\max \eta_i$, $\min \eta_i$

Table 2. Comparison of the each algorithm for various $A \in \mathbb{R}^{n\times n}$

n	$\max \eta_i^{(1)}$	$\min \eta_i^{(1)}$	$\max \eta_i^{(2)}$	$\min \eta_i^{(2)}$	$\max \eta_i^{(3)}$	$\min \eta_i^{(3)}$	δ
100	1.593e-11	1.546e-12	9.810e-11	6.697e-12	1.477e-12	2.800e-14	6.123e-12
250	1.359e-10	9.821e-12	1.322e-09	5.031e-11	7.484e-12	1.100e-13	4.588e-11
500	7.007e-10	4.489e-11	9.953e-09	2.482e-10	1.472e-11	2.659e-13	1.436e-10
1000	3.683e-09	2.843e-10	7.707e-08	1.324e-09	6.017e-11	8.529e-13	8.266e-10
1500	9.857e-09	6.575e-10	2.553e-07	3.378e-09	1.587e-10	1.344e-12	2.621e-09
2000	2.926e-08	1.331e-09	6.094e-07	6.761e-09	4.978e-10	2.254e-12	6.517e-09

n	t_1	t_2	t_3	t_δ	t_{eig}	t_{vec}
100	0.016	0.015	0.016	0.016	0.006	0.010
250	0.125	0.094	0.094	0.094	0.031	0.125
500	0.813	0.437	0.531	0.484	0.203	0.969
1000	5.609	2.500	3.187	3.062	1.688	6.921
1500	17.47	6.984	9.781	9.469	5.281	22.70
2000	40.61	15.41	22.19	21.66	12.27	53.64

for $1 \leq i \leq n$, δ and t defined in Table 1. In addition, t_{eig} and t_{vec} are also displayed. Here, (29) holds in all of examples within Table 2. By Table 2, it can be seen that Algorithm 3 supplies smaller error bounds than those by the algorithms proposed in [6] and [5]. Since (29) held in all of examples within Table 2, it held that $\max \eta_i^{(3)} \leq \delta$. This result identifies Theorem 3. It can also be seen that Algorithm 3 was faster than the algorithm proposed in [6]. This result identifies the fact that the computational cost of Algorithm 3 is $5n^3$ flops while the computational cost of the algorithm proposed in [6] is $8n^3$ flops. Moreover, Algorithm 3 is a little slower than the algorithm proposed in [5]. This results from the fact that the computational cost of the algorithm proposed in [5] is $3n^3$ flops. Additionally the speed of Algorithm 3 was approximately equal to that of Algorithm 1. This result identifies the consideration presented in Section 3.

Table 3. Comparison of the each algorithm for various $A \in \mathbb{R}^{n\times n}$ whose eigenvalues are partially clustered

n	$\max \eta_i^{(1)}$	$\min \eta_i^{(1)}$	$\max \eta_i^{(2)}$	$\min \eta_i^{(2)}$	$\max \eta_i^{(3)}$	$\min \eta_i^{(3)}$	δ
100	7.297e-14	2.454e-14	2.265e-12	7.437e-13	5.703e-15	1.651e-15	4.131e-14
250	2.497e-13	8.495e-14	1.404e-11	4.982e-12	1.242e-14	4.298e-15	1.518e-13
500	6.118e-13	2.039e-13	5.577e-11	1.910e-11	2.377e-14	8.446e-15	4.211e-13
1000	1.506e-12	4.721e-13	2.201e-10	7.494e-11	4.654e-14	1.676e-14	1.150e-12
1500	2.743e-12	9.052e-13	5.113e-10	1.860e-10	2.161e-12	2.714e-14	2.161e-12
2000	3.907e-12	1.173e-12	8.662e-10	2.909e-10	3.144e-12	3.190e-14	3.144e-12

n	m	t_1	t_2	t_3	t_δ	t_{eig}	t_{vec}
100	3	0.016	0.015	0.016	0.015	0.006	0.010
250	4	0.125	0.094	0.094	0.093	0.015	0.110
500	4	0.812	0.453	0.547	0.500	0.204	0.749
1000	5	5.468	2.500	3.234	3.078	1.672	5.063
1500	5	17.36	7.031	9.829	9.485	5.282	18.23
2000	5	40.48	15.41	22.42	21.75	12.22	40.08

At last, let A be a real symmetric $n \times n$ matrix whose eigenvalues are partially clustered at regular interval 1e-13. Table 3 shows similar results to Table 2. Here, m in Table 3 is the number of Λ_i in **Step 3** within Algorithm 3. When $n = 100$, (29) held. When $n = 250, 500$ and 1000, (29) did not hold and (30) held. When $n = 1500$ and 2000, neither (29) nor (30) held. We are able to confirm that $\max \eta_i^{(3)} = \delta$ held for $n = 1500$ and 2000 because neither (29) nor (30) held. The other tendencies about the error bounds and the computing time were similar to Table 2.

In conclusion, it turns out that we can efficiently obtain tighter error bounds of respective eigenvalues by the proposed algorithm (Algorithm 3) with almost the same computing time as that of Rump's algorithm.

References

1. ANSI/IEEE: IEEE Standard for Binary Floating-Point Arithmetic: ANSI/IEEE Std 754-1985. IEEE, New York, 1985
2. Golub, G. H., van Loan, C. F.: Matrix Computations, Third edition. The Johns Hopkins University Press, Baltimore and London, 1996
3. Higham, N. J.: Accuracy and Stability of Numerical Algorithms, Second edition. SIAM Publications, Philadelphia, 2002
4. Ipsen, I. C. F.: Relative Perturbation Bounds for Matrix Eigenvalues and Singular Values, Acta Numerica **7**, Cambridge University Press, Cambridge, 151–201, 1998
5. Miyajima, S., Ogita, T., Ozaki, K., Oishi, S: Fast error estimation for eigenvalues of symmetric matrix without directed rounding. Proc. 2004 International Symposium on Nonlinear Theory and its Applications, Fukuoka, Japan, 2004, 167–170
6. Oishi, S.: Fast enclosure of matrix eigenvalues and singular values via rounding mode controlled computation. Linear Alg. Appl. **324** (2001) 133–146
7. Parlett, B. N.: The Symmetric Eigenvalue Problem, Classics in Applied Mathematics **20**, SIAM Publications, Philadelphia, 1997
8. Rump, S. M.: Computational error bounds for multiple or nearly multiple eigenvalues, Linear Alg. Appl. **324** (2001) 209–226
9. Rump, S. M.: private communication. 2004
10. Rump, S. M., Zemke, J.: On eigenvector bounds. BIT **43** (2004) 823–837
11. Stewart, G. W., Sun, J.-G.: Matrix Perturbation Theory, Academic Press, New York, 1990
12. The MathWorks Inc.: MATLAB User's Guide Version 7. 2004
13. Wilkinson, J. H.: Rigorous error bounds for computed eigensystem. Computer J. **4** (1961) 230–241
14. Yamamoto, T.: Error bounds for computed eigenvelues and eigenvectors. Numer. Math. **34** (1980) 189–199

Towards More Accurate Separation Bounds of Empirical Polynomials II

Kosaku Nagasaka

Faculty of Human Development, Kobe University, Japan
nagasaka@main.h.kobe-u.ac.jp

Abstract. We study the problem of bounding a polynomial which is absolutely irreducible, away from polynomials which are not absolutely irreducible. These separation bounds are useful for testing whether an empirical polynomial is absolutely irreducible or not, for the given tolerance or error bound of its coefficients. In the former paper, we studied some improvements on Kaltofen and May's method which finds applicable separation bounds using an absolute irreducibility criterion due to Ruppert. In this paper, we study the similar improvements on the method using the criterion due to Gao and Rodrigues for sparse polynomials satisfying Newton polytope conditions, by which we are able to find more accurate separation bounds, for such bivariate polynomials. We also discuss a concept of separation bound continuations for both dense and sparse polynomials.

1 Introduction

We consider numerical polynomials with certain tolerances, including empirical polynomials with error bounds on its coefficients, which are useful for applied computations of polynomials. We have to use completely different algorithms from the conventional algorithms since we have to take care of their errors on coefficients and have to guarantee the results within the given tolerances.

In this paper and the former paper [1], we focus on testing absolute irreducibilities of such polynomials, hence we consider the following problem.

Problem 1. For the given polynomial $f \in \mathbb{C}[x,y]$ which is absolutely irreducible, compute the largest value $B(f) \in \mathbb{R}_{>0}$ such that all $\tilde{f} \in \mathbb{C}[x,y]$ with $\|f-\tilde{f}\|_2 < B(f)$ (and $\deg(\tilde{f}) \leq \deg(f)$) must remain absolutely irreducible. $\triangleleft$

This problem is studied by Kaltofen [2], however its separation bound is too small. The first applicable bound is given by the author [3], using an absolute irreducibility criterion due to Sasaki [4], and slightly improved by the author [5]. In ISSAC'03, Kaltofen and May [6] studied an efficient method using an absolute irreducibility criterion due to Ruppert [7], and a similar criterion due to Gao and Rodrigues [8] for sparse polynomials. The former paper [1] gave some improvements on Kaltofen and May's method due to Ruppert. Similar improvements on

[1] This research is partly helped by Grants-in-Aid of MEXT, JAPAN, #16700016.

their method due to Gao and Rodrigues can be available partly. This is one of main topics in this paper. Hence, the problem becomes the following.

Problem 2. For the given polynomial $f \in \mathbb{C}[x, y]$ which is absolutely irreducible, compute the largest value $\bar{B}(f) \in \mathbb{R}_{>0}$ such that all $\tilde{f} \in \mathbb{C}[x, y]$ satisfying $\mathcal{P}(\tilde{f}) \subseteq \mathcal{P}(f)$ with $\| f - \tilde{f} \|_2 < \bar{B}(f)$ must remain absolutely irreducible, where $\mathcal{P}(p)$ means the Newton polytope of a polynomial p. $\triangleleft$

This is better for the case where we limit the changeable terms to being in the polytope. We note that the Newton polytope of a polynomial $p = \sum_{i,j} a_{i,j} x^i y^j$ is defined as the convex hull in the Euclidean plane $\mathbb{R}^2$ of the exponent vectors (i, j) of all the nonzero terms of p.

Example 1. Let $f(x, y)$ be the following irreducible polynomial in x and y.

$$f(x, y) = (x^2 + yx + 2y - 1)(x^3 + y^2 x - y + 7) + 0.2x.$$

We have $B(f)/\|f\|_2 = 3.867 \times 10^{-5}$, by Kaltofen and May's algorithm. Hence, any polynomial which is included in ε-neighborhood of $f(x, y)$ in 2-norm, is still absolutely irreducible, where $\varepsilon = 3.867 \times 10^{-5}$. This bound can be optimized to 4.247×10^{-5} by the improved method [1]. We note that this polynomial can be factored approximately with the backward errors 7.531×10^{-4} [3] and 1.025×10^{-3} [9]. For the problem 2, we have $\bar{B}(f)/\|f\|_2 = 1.349 \times 10^{-4}$. We note that we have $\bar{B}(f) \leq B(f)$ for any polynomial f, since the all changeable terms in the sense of Problem 2 are included in those terms of Problem 1. $\triangleleft$

The contribution of this paper is the following two points; 1) refining the Kaltofen and May's algorithm due to Gao and Rodrigues and finding more accurate separation bounds, 2) a discussion about a concept of separation bound continuations for both dense and sparse polynomials.

2 Original Method

Kaltofen and May's method mainly uses the following absolute irreducibility criterion due to Ruppert [7]. For the given polynomial, consider the following differential equation w.r.t. unknown polynomials g and h.

$$f \frac{\partial g}{\partial y} - g \frac{\partial f}{\partial y} + h \frac{\partial f}{\partial x} - f \frac{\partial h}{\partial x} = 0, \quad g, h \in \mathbb{C}[x, y], \tag{1}$$

$$\deg_x g \leq \deg_x f - 1, \ \deg_y g \leq \deg_y f, \ \deg_x h \leq \deg_x f, \ \deg_y h \leq \deg_y f - 2.$$

The criterion is that $f(x, y)$ is absolutely irreducible if and only if this differential equation (1) does not have any non-trivial solutions.

Their method uses matrix representations of absolute irreducibility criteria, and check whether those matrices are of certain ranks or not. They use the following matrix, for the above criterion, considering the above differential equation w.r.t. g and h as a linear system w.r.t. unknown coefficients of g and h.

$$\begin{pmatrix}
G_n & 0 & \cdots & 0 & 0\cdot H_n & 0 & \cdots & 0 & 0 \\
G_{n-1} & G_n & \ddots & \vdots & -H_{n-1} & H_n & \ddots & \vdots & \vdots \\
\vdots & G_{n-1} & \ddots & 0 & \vdots & 0\cdot H_{n-1} & \ddots & 0 & \vdots \\
G_1 & \vdots & \ddots & G_n & (1-n)H_1 & \vdots & \ddots & (n-1)H_n & 0 \\
G_0 & G_1 & \ddots & G_{n-1} & -nH_0 & (2-n)H_1 & \ddots & (n-2)H_{n-1} & nH_n \\
0 & G_0 & \ddots & \vdots & 0 & (1-n)H_0 & \ddots & \vdots & (n-1)H_{n-1} \\
\vdots & \ddots & \ddots & G_1 & \vdots & 0 & \ddots & 0\cdot H_1 & \vdots \\
0 & \cdots & 0 & G_0 & 0 & \cdots & 0 & -H_0 & H_1
\end{pmatrix}$$

$$G_i = \begin{pmatrix}
0 & 0 & \cdots & 0 & 0 & 0 \\
c_{i,m-1} & -c_{i,m} & \ddots & \vdots & \vdots & 0 \\
2c_{i,m-2} & 0 & \ddots & 0 & \vdots & \vdots \\
\vdots & c_{i,m-2} & \ddots & (2-m)\,c_{i,m} & 0 & \vdots \\
\vdots & \vdots & \ddots & \vdots & (1-m)\,c_{i,m} & 0 \\
m\,c_{i,0} & \vdots & \ddots & \vdots & \vdots & -m\,c_{i,m} \\
0 & (m-1)\,c_{i,0} & \ddots & 0 & \vdots & \vdots \\
\vdots & 0 & \ddots & c_{i,1} & -c_{i,2} & \vdots \\
0 & \vdots & \ddots & 2c_{i,0} & 0 & -2c_{i,2} \\
0 & 0 & \cdots & 0 & c_{i,0} & -c_{i,1}
\end{pmatrix}, \quad H_i = \begin{pmatrix}
0 & \cdots & 0 \\
c_{i,m} & \ddots & \vdots \\
c_{i,m-1} & \ddots & 0 \\
\vdots & \ddots & c_{i,m} \\
c_{i,1} & \ddots & c_{i,m-1} \\
c_{i,0} & \ddots & \vdots \\
 & \ddots & c_{i,1} \\
0 & & c_{i,0}
\end{pmatrix}$$

Fig. 1. Ruppert matrix $R(f)$

Let $R(f)$ be the coefficient matrix of the linear system as in the figure 1, where the block matrices G_i and H_i are the matrices of sizes $2m \times (m+1)$ and $2m \times (m-1)$, respectively, where the given polynomial be

$$f = \sum_{i=0}^{n} \sum_{j=0}^{m} c_{i,j} x^i y^j, \quad c_{i,j} \in \mathbb{C}.$$

We call $R(f)$ the Ruppert matrix. The size of Ruppert matrix $R(f)$ is $(4nm) \times (2nm + m - 1)$ where $n = \deg_x(f)$ and $m = \deg_y(f)$.

The original expressions of separation bounds in Kaltofen and May's algorithm [6] are the following $B_\alpha(f)$ and $B_\beta(f)$. We note that $B_\alpha(f)$ is a lower bound of $B_\beta(f)$ by bounding the largest coefficient of $\|R(\varphi)\|_F^2$ in $B_\beta(f)$, where $\|A\|_F$ denotes the Frobenius norm (the square root of the sum of squares of all the elements) of matrix A.

$$B_\alpha(f) = \frac{\sigma(R(f))}{\max\{n,m\}\sqrt{2nm-n}}, \; B_\beta(f) = \frac{\sigma(R(f))}{\sqrt{(\text{the largest coef. of } \|R(\varphi)\|_F^2)}},$$

where $R(\varphi)$ denotes $R(f)$ calculated by treating $c_{i,j}$ as variables and $\sigma(A)$ denotes the $(2nm + m - 1)$-th largest singular value of matrix A.

3 Previous Work

In the former article [1], we decomposed $R(f)$ to integer matrices and complex coefficients parts, and gave some improvements using those matrices.

We refer the former results, briefly. The Ruppert matrix can be written as

$$R(f) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}c_{i,j}, \quad R_{i,j} \in \mathbb{Z}^{(4nm)\times(2nm+m-1)}, \tag{2}$$

where each elements of $R_{i,j}$ is an integer coefficient generated by differentiating polynomials, and $R_{i,j}$ has the same shape as $R(f)$ but whose elements are different. Then, the expressions of separation bounds can be refined as the following expression, by Lemma 1 in the former paper.

$$B(f) = \sqrt{6}\,\sigma(R(f))/\sqrt{n(m(m+1)(2m+1)+(m-1)(n+1)(2n+1))}. \tag{3}$$

3.1 Improvement Strategy

We refer the strategy of the former paper [1], improving the original method of Kaltofen and May due to the Ruppert.

The method uses the absolute irreducibility criteria as a necessary condition which the given polynomial is absolutely irreducible. In the Kaltofen and May's algorithm, $\sigma(R(f))$ is considered as a threshold whether the differential equation (or the linear system) (1) has non-trivial solutions or not. In this point of view, to determine that the differential equation does not have non-trivial solutions, corresponding to that the given polynomial is absolutely irreducible, we do not need to use all the constraint equations w.r.t. unknown coefficients of polynomials g and h, since the corresponding linear system is over-determined. We can lessen the number of constraint equations appeared in the Ruppert matrix $R(f)$, without decreasing its matrix rank.

We note that removing rows (constraint equations) may decrease the numerator of the expression (3) and may decrease the denominator depending on the elements of $R_{i,j}$. Hence, depending on variations of the numerator and denominator, $R(f)$ changes and it can be larger if we choose suitable rows.

As in the former paper, we define the following "drop" notations for removing rows from a matrix, which are corresponding to removing constraint equations.

$$\mathrm{drop}_i(A) = (\boldsymbol{a}_1,\ldots,\boldsymbol{a}_{i-1},\boldsymbol{0},\boldsymbol{a}_{i+1},\ldots,\boldsymbol{a}_{k_1})^t,\ A = (\boldsymbol{a}_1,\ldots,\boldsymbol{a}_{k_1})^t \in \mathbb{C}^{k_1\times k_2},$$

$$R^{(k)}(f) = \mathrm{drop}_{d_k}(\cdots(\mathrm{drop}_{d_1}(R(f)))),\ \ R_{i,j}^{(k)} = \mathrm{drop}_{d_k}(\cdots(\mathrm{drop}_{d_1}(R_{i,j}))),$$

where $d_1,\ldots,d_k$ are indices of rows removed from the given matrix.

Improving the original method now becomes the following problem.

Problem 3. Find an integer k, row indices d_1, ..., d_k to be removed, and the following separation bound $B^{(k)}(f) > B(f)$.

$$B^{(k)}(f) = \sigma(R^{(k)}(f))/\max_{i,j}\|R_{i,j}^{(k)}\|_F\ . \qquad \triangleleft$$

Lemma 1 (Lemma 2 in [1]). *We have to remove at least* 2 *rows* ($k = 2$) *from the Ruppert matrix for finding more accurate separation bounds satisfying* $B^{(k)}(f) > B(f)$*, For* $k = 2$*, rows to be removed from the matrix, must satisfy*

$$\begin{cases} d_1 = 2md_x + d_y \ (0 \le d_x \le n-1 \ \wedge \ d_y = m+1), \\ d_2 = 2md_x + d_y \ (n \le d_x \le 2n-1 \ \wedge \ d_y = m+1) \end{cases}$$
$$or \begin{cases} d_1 = 2md_x + d_y \ (d_x = n \ \wedge \ 2 \le d_y \le m), \\ d_2 = 2md_x + d_y \ (d_x = n \ \wedge \ m+2 \le d_y \le 2m). \end{cases} \qquad \triangleleft$$

By Lemma 1, the simple algorithm was introduced, which give us about 1.6% more accurate separation bounds, according to the experimental result in the former paper. We note that "removing multiple rows" versions of the algorithm were also introduced in the paper.

4 Newton Polytope Version

Kaltofen and May also argued briefly the method using the following criterion due to Gao and Rodrigues [8] which is effective for factoring sparse polynomials. For the given polynomial, consider the following differential equation w.r.t. unknown polynomials g and h in $\mathbb{C}[x, y]$.

$$f\frac{\partial g}{\partial y} - g\frac{\partial f}{\partial y} + h\frac{\partial f}{\partial x} - f\frac{\partial h}{\partial x} = 0, \ \ \mathcal{P}(xg) \subseteq \mathcal{P}(f) \text{ and } \mathcal{P}(yh) \subseteq \mathcal{P}(f). \qquad (4)$$

The criterion that the given polynomial is absolutely irreducible is a little bit different from the Ruppert criterion. Let $\mathcal{R}(f)$ be the coefficient matrix of the linear system of the above differential equation (4) w.r.t. unknown coefficients of polynomials g and h. We call $\mathcal{R}(f)$ the sparse Ruppert matrix. Polynomials g and h do not have the same forms as in the differential equation (1) by Ruppert, hence, for sparse polynomials, the size of sparse Ruppert matrix $\mathcal{R}(f)$ is less than the size of Ruppert matrix $R(f)$. The figure of the sparse Ruppert matrix is depending on the Newton polytope of the given polynomial and we can not show its general form. For easiness of discussions, we define the skeleton of the sparse Ruppert matrix $\bar{R}(f)$, with full terms of g and h, as in the figure 1, where the block matrices G_i and H_i are the matrices of sizes $2m \times (m+1)$ and $2m \times m$, respectively, as in the figure 2. The size of the skeleton matrix $R(f)$ is $(4nm) \times (2nm + n + m)$. We note that 1) the only difference between $R(f)$ and $\bar{R}(f)$ is on the block matrix H_i, 2) an actual sparse Ruppert matrix $\mathcal{R}(f)$ can be generated by replacing all elements with zeros, on some columns corresponding to unnecessary terms of polynomials g and h by the condition due to the Newton polytope of $f(x, y)$, or by removing such columns.

The criterion is that $f(x, y)$ is absolutely irreducible if and only if the sparse Ruppert matrix $\mathcal{R}(f)$ has the rank $\rho-1$, where ρ denotes the number of unknown coefficients of polynomials g and h. We note that Problem 2 is corresponding to this criterion, and contributions of this paper are mainly for this problem. We have the following separation bound $\bar{B}_\beta(f)$, by the same way of the paper [6].

$$\bar{B}_\beta(f) = \bar{\sigma}(\mathcal{R}(f))/\sqrt{(\text{the largest coefficient of } \ \|\mathcal{R}(\varphi)\|_F^2)},$$

$$G_i = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & 0 \\ c_{i,m-1} & -c_{i,m} & \ddots & \vdots & \vdots & 0 \\ 2c_{i,m-2} & 0 & \ddots & 0 & \vdots & \vdots \\ \vdots & c_{i,m-2} & \ddots & (2-m) \times c_{i,m} & 0 & \vdots \\ \vdots & \vdots & \ddots & \vdots & (1-m)\, c_{i,m} & 0 \\ m\, c_{i,0} & \vdots & \ddots & \vdots & \vdots & -m\, c_{i,m} \\ 0 & (m-1)\, c_{i,0} & \ddots & 0 & \vdots & \vdots \\ \vdots & 0 & \ddots & c_{i,1} & -c_{i,2} & \vdots \\ 0 & \vdots & \ddots & 2c_{i,0} & 0 & -2c_{i,2} \\ 0 & 0 & \cdots & 0 & c_{i,0} & -c_{i,1} \end{pmatrix}, \quad H_i = \begin{pmatrix} c_{i,m} & 0 & \cdots & 0 \\ c_{i,m-1} & c_{i,m} & \ddots & \vdots \\ c_{i,m-2} & c_{i,m-1} & \ddots & 0 \\ \vdots & \vdots & \ddots & c_{i,m} \\ c_{i,0} & c_{i,1} & \ddots & c_{i,m-1} \\ 0 & c_{i,0} & \ddots & \vdots \\ \vdots & \ddots & \ddots & c_{i,1} \\ 0 & \cdots & & c_{i,0} \end{pmatrix}$$

Fig. 2. Block matrices of skeleton matrix $\bar{R}(f)$

where $\bar{\sigma}(A)$ denotes the $(\rho-1)$-th largest singular value of matrix A. In the rest of this paper, we discuss the similar refining of $\bar{B}_\beta(f)$ as in the former paper [1].

4.1 Integer Matrices

We decompose $\mathcal{R}(f)$ and $\bar{R}(f)$ to integer matrices and complex coefficients parts, as in the previous section. These matrices can be written as

$$\mathcal{R}(f) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathcal{R}_{i,j} c_{i,j}, \quad \bar{R}(f) = \sum_{i=0}^{n} \sum_{j=0}^{m} \bar{R}_{i,j} c_{i,j},$$

where each elements of $\mathcal{R}_{i,j}$ and $\bar{R}_{i,j}$ is an integer coefficient generated by differentiating polynomials, and $\mathcal{R}_{i,j}$ and $\bar{R}_{i,j}$ have the same shape of $\mathcal{R}(f)$ and $\bar{R}(f)$, respectively, but all the elements are defined as in the figure 3 where $\delta_{i,j}$ denotes Kronecker delta. These integer matrices have the following properties similar to those of the integer matrices of the Ruppert matrix.

Lemma 2. *We have*

$$\max_{i,j} \|\bar{R}_{i,j}\|_F^2 = nm((2n+1)(n+1)+(2m+1)(m+1))/6. \qquad \triangleleft$$

Proof. The same way as in Lemma 1 in [1]. □

Corollary 1. *We have the following equality.*

$$\max_{i,j} \|\bar{R}_{i,j}\|_F = \|\bar{R}_{n,m}\|_F = \|\bar{R}_{n,0}\|_F = \|\bar{R}_{0,m}\|_F = \|\bar{R}_{0,0}\|_F\ . \qquad \triangleleft$$

$$\begin{pmatrix}
\delta_{i,n}G_n & \cdots & 0 & 0\cdot\delta_{i,n}H_n & 0 & \cdots & 0 & 0 \\
\delta_{i,n-1}G_{n-1} & \ddots & \vdots & -\delta_{i,n-1}H_{n-1} & \delta_{i,n}H_n & \ddots & \vdots & \vdots \\
\vdots & \ddots & 0 & \vdots & 0\cdot\delta_{i,n-1}H_{n-1} & \ddots & 0 & \vdots \\
\delta_{i,1}G_1 & \ddots & \delta_{i,n}G_n & (1-n)\delta_{i,1}H_1 & \vdots & \ddots & (n-1)\delta_{i,n}H_n & 0 \\
\delta_{i,0}G_0 & \ddots & \delta_{i,n-1}G_{n-1} & -n\delta_{i,0}H_0 & (2-n)\delta_{i,1}H_1 & \ddots & (n-2)\delta_{i,n-1}\times H_{n-1} & nH_n \\
0 & \ddots & \vdots & 0 & (1-n)\delta_{i,0}H_0 & \ddots & \vdots & (n-1)\delta_{i,n-1}\times H_{n-1} \\
\vdots & \ddots & \delta_{i,1}G_1 & \vdots & 0 & \ddots & 0\cdot\delta_{i,1}H_1 & \vdots \\
0 & \cdots & \delta_{i,0}G_0 & 0 & \cdots & 0 & -\delta_{i,0}H_0 & \delta_{i,1}H_1
\end{pmatrix}$$

$$G_i = \begin{pmatrix}
0 & 0 & \cdots & 0 & 0 & 0 \\
\delta_{j,m-1} & -\delta_{j,m} & \ddots & \vdots & \vdots & 0 \\
2\delta_{j,m-2} & 0 & \ddots & 0 & \vdots & \vdots \\
\vdots & \delta_{j,m-2} & \ddots & (2-m)\times\delta_{j,m} & 0 & \vdots \\
\vdots & \vdots & \ddots & \vdots & (1-m)\,\delta_{j,m} & 0 \\
m\,\delta_{j,0} & \vdots & \ddots & \vdots & \vdots & -m\,\delta_{j,m} \\
0 & (m-1)\,\delta_{j,0} & \ddots & 0 & \vdots & \vdots \\
\vdots & 0 & \ddots & \delta_{j,1} & -\delta_{j,2} & \vdots \\
0 & \vdots & \ddots & 2\delta_{j,0} & 0 & -2\delta_{j,2} \\
0 & 0 & \cdots & 0 & \delta_{j,0} & -\delta_{j,1}
\end{pmatrix}, \quad
H_i = \begin{pmatrix}
\delta_{j,m} & 0 & \cdots & 0 \\
\delta_{j,m-1} & \delta_{j,m} & \ddots & \vdots \\
\delta_{j,m-2} & \delta_{j,m-1} & \ddots & 0 \\
\vdots & \vdots & \ddots & \delta_{j,m} \\
\delta_{j,0} & \delta_{j,1} & \ddots & \delta_{j,m-1} \\
0 & \delta_{j,0} & \ddots & \vdots \\
\vdots & \ddots & \ddots & \delta_{j,1} \\
0 & \cdots & & \delta_{j,0}
\end{pmatrix}$$

Fig. 3. Integer matrix $R_{i,j}$

Remark 1. Using the above lemma and corollary, we can rewrite the expression $\bar{B}_\beta(f)$ as in the former paper, however, it is useless since an actual sparse Ruppert matrix which does not have some columns corresponding to unnecessary terms of polynomials g and h by the Newton polytope of $f(x,y)$, and separation bounds based on $\bar{R}(f)$ may be larger than those of an actual sparse Ruppert matrix $\mathcal{R}(f)$. Hence, we have to use the expression $\bar{B}_\beta(f)$ still. This is different from the Ruppert matrix case. $\lhd$

The following lemma helps us to calculate the largest coefficient of $\|\mathcal{R}(\varphi)\|_F^2$ which is appeared in the denominator of $\bar{B}_\beta(f)$.

Lemma 3. *We have the following equality.*

$$\max_{i,j} \|\mathcal{R}_{i,j}\|_F = \max\{\|\mathcal{R}_{n,m}\|_F, \|\mathcal{R}_{n,0}\|_F, \|\mathcal{R}_{0,m}\|_F, \|\mathcal{R}_{0,0}\|_F\}. \qquad \lhd$$

Proof. Since we can construct $\mathcal{R}_{i,j}$ by removing some columns from $\bar{R}_{i,j}$ (or replacing them with zeros), we only have to prove that Corollary 1 is still valid after removing columns. We focus only on the index j and consider the left hand side of $\bar{R}_{i,j}$ formed by block matrices G_i and the the right hand side of $\bar{R}_{i,j}$ formed by block matrices H_i separately.

For the right hand side, each sum of squares of elements corresponding to an index j on each column has the same Frobenius norm. Hence, removing columns on the right hand side does not affect the equality of Corollary 1. Therefore, we only have to show that: the largest coefficients of $c_{i,m}$ and $c_{i,0}$ of the Frobenius norm of G_i is the largest coefficient among $c_{i,j}$ after removing.

Let $\Delta_{k,0}$ and $\Delta_{k,m}$ be differences between the coefficients of $c_{i,0}$ and $c_{i,m}$ and $c_{i,m-\kappa}$ of $\|G_i\|_F^2$ on the $k+1$-th column, respectively. We have

$$\begin{aligned} \Delta_{k,0} &= (m-k)^2 - (m-k-m+\kappa)^2 = (2k-\kappa-m)(\kappa-m), \\ \Delta_{k,m} &= (-k)^2 - (m-k-m+\kappa)^2 \quad\ \ = (2k-\kappa)\kappa. \end{aligned}$$

Let $\mathcal{I}$ be the set of column indices of the rest columns after removing. We suppose that the lemma is not valid and $c_{i,m-\kappa}$ has the largest coefficient. We have

$$\sum_{k\in\mathcal{I}} \Delta_{k,0} = (\kappa-m)\sum_{k\in\mathcal{I}}(2k-\kappa-m) < 0, \quad \sum_{k\in\mathcal{I}} \Delta_{k,m} = \kappa\sum_{k\in\mathcal{I}}(2k-\kappa) < 0.$$

Since $\kappa - m$ is not positive and κ is not negative, we have

$$\sum_{k\in\mathcal{I}}(2k-\kappa-m) = \sum_{k\in\mathcal{I}}(2k-\kappa) - \#\mathcal{I}m > 0, \quad \sum_{k\in\mathcal{I}}(2k-\kappa) < 0,$$

where $\#\mathcal{I}$ denotes the number of elements of the set $\mathcal{I}$. This leads a contradiction. Therefore the lemma is valid. We note that we can prove for the index i by the similar way even if it not necessary for the proof. □

By Lemma 3, we have the following separation bound.

$$\bar{B}(f) = \bar{\sigma}(\mathcal{R}(f))/\max\{\|\mathcal{R}_{n,m}\|_F, \|\mathcal{R}_{n,0}\|_F, \|\mathcal{R}_{0,m}\|_F, \|\mathcal{R}_{0,0}\|_F\}.$$

4.2 Improvement Strategy

For the sparse Ruppert matrix, the improvement strategy of the former paper is still applicable. Hence, the aim of this subsection is the following problem.

Problem 4. Find an integer k, indices $d_1, \ldots, d_k$ to be removed, and the following separation bound $\bar{B}^{(k)}(f) > \bar{B}(f)$.

$$\bar{B}^{(k)}(f) = \bar{\sigma}(\mathcal{R}^{(k)}(f))/\max_{i,j} \|\mathcal{R}_{i,j}^{(k)}\|_F \ . \qquad \triangleleft$$

- Removing Two Rows - For the sparse Ruppert matrix, we still consider "removing two rows from the matrix" even though the important corollary in [1] is not valid and we have only Lemma 3. Because even for such cases, we may have to remove rows providing that $\|\mathcal{R}_{n,m}\|_F$, $\|\mathcal{R}_{n,0}\|_F$, $\|\mathcal{R}_{0,m}\|_F$ and $\|\mathcal{R}_{0,0}\|_F$ become smaller and $\bar{B}^{(k)}(f) > \bar{B}(f)$, depending on $\mathcal{R}(f)$. Therefore, we follow the same discussion. We consider variations of $\|\bar{R}_{i,j}\|_F$, provided by removing a $(2md_x + d_y)$-th row from $\bar{R}(f)$, satisfying $0 \le d_x \le 2n-1$ and $1 \le d_y \le 2m$.

Let Δ_G be the square of Frobenius norm of variations of the left hand side part of $\bar{R}_{i,j}$, corresponding to G_i and Δ_H be that of the right hand side part of $\bar{R}_{i,j}$, corresponding to H_i. We have

$$\|\mathrm{drop}_{2md_x+d_y}(\bar{R}_{i,j})\|_F^2 = \|\bar{R}_{i,j}\|_F^2 - \Delta_G - \Delta_H.$$

By the same way in the former paper, we have the following relations that are slightly different from those of the Ruppert matrix.

$$\Delta_G = \begin{cases} 0 & (i < n - d_x) \vee (2n - d_x - 1 < i) \vee \\ & \quad (j < m - d_y + 1) \vee (2m + 1 - d_y < j) \\ (2m + 1 - d_y - 2j)^2 & \text{otherwise} \end{cases} \tag{5}$$

$$\Delta_H = \begin{cases} 0 & (i < n - d_x) \vee (2n - d_x < i) \vee \\ & \quad (j < m - d_y + 1) \vee (2m - d_y < j) \\ (2i + d_x - 2n)^2 & \text{otherwise} \end{cases} \tag{6}$$

Lemma 4. *We may have to remove at least* 2 *rows* ($k = 2$) *from the sparse Ruppert matrix for finding more accurate separation bound satisfying* $\bar{B}^{(k)}(f) > \bar{B}(f)$. *For* $k = 2$, *rows to be removed from the matrix, should satisfy*

$$\begin{cases} d_1 = 2md_x + d_y \ (0 \le d_x \le n - 1 \ \wedge \ d_y = m + 1), \\ d_2 = 2md_x + d_y \ (n \le d_x \le 2n - 1 \ \wedge \ d_y = m + 1) \end{cases}$$
$$or \begin{cases} d_1 = 2md_x + d_y \ (d_x = n \ \wedge \ 1 \le d_y \le m), \\ d_2 = 2md_x + d_y \ (d_x = n \ \wedge \ m + 1 \le d_y \le 2m). \end{cases} \quad \triangleleft$$

Proof. The same way as in Lemma 2 in [1]. □

We note that removing only one row has possibility to satisfy $\bar{B}^{(1)}(f) > \bar{B}(f)$, since we have only Lemma 3 for the sparse Ruppert matrix. However, the above lemma guarantees that removing such two rows must decrease $\max_{i,j} \|\mathcal{R}_{i,j}\|_F = \max\{\|\mathcal{R}_{n,m}\|_F, \|\mathcal{R}_{n,0}\|_F, \|\mathcal{R}_{0,m}\|_F, \|\mathcal{R}_{0,0}\|_F\}$.

By Lemma 4, we have the following simple algorithm which give us about 1.3% more accurate separation bounds, according to our experimental result.

Algorithm 1. (Removing Two Rows Sparse Version)
Input: a bivariate polynomial $f(x, y)$, *Output*: a separation bound $\bar{B}(f)$

Step 1 Construct sparse Ruppert matrix $\mathcal{R}(f)$.
Step 2 For all index pairs d_1 and d_2 in Lemma 4, compute separation bounds, and let the best separation bound be $\bar{B}^{(2)}(f)$.
Step 3 Output the separation bound $\bar{B}^{(2)}(f)$ and finish the algorithm. $\triangleleft$

- Removing Multiple Rows - For the Ruppert matrix, in the former paper, by the lemma which guarantees Lemma 3 after removing rows, the algorithms removing multiple rows were introduced. For the sparse Ruppert matrix, such a lemma does not exist since an actual sparse Ruppert matrix does not have a lots of columns and removing rows easily breaks Lemma 3. However, we can use the similar algorithms though they are not effective as before.

```
Algorithm 2. (Early Termination Algorithm Sparse Version)
Input: a bivariate polynomial $f(x,y)$,        Output: a separation bound $\bar{B}(f)$
Step 1 Construct sparse Ruppert matrix $\mathcal{R}(f)$ and put $k=1$.
Step 2 Compute contributing ratios of each rows of $\mathcal{R}(f)$.
Step 3 Construct all the index pairs $d_{2k-1}$ and $d_{2k}$ as in Lemma 4.
Step 4 For each index pairs constructed in Step 3, compute separation bounds
    with $d_1, d_2, \ldots, d_{2k}$, by ascending order of sums of contributing ratios, until
    an index pair for which a separation bound does not become better than that
    of a previous group twice, and let the best separation bound be $\bar{B}^{(2k)}(f)$.
Step 5 If $\bar{B}^{(2k-2)}(f) \leq \bar{B}^{(2k)}(f)$ then put $k=k+1$ and goto Step 3.
Step 6 Output the separation bound $\bar{B}^{(2k-2)}(f)$ and finish the algorithm.  ◁
```

We use Euclidean norms of corresponding row vectors of the Moore-Penrose type pseudo inverse of the transpose of $\mathcal{R}(f)$ as the contributing ratios (see [1]).

Example 2. For the polynomial in the example 1, the algorithms 1 and 2 output $\bar{B}(f) = 1.420 \times 10^{-4}$ and $\bar{B}(f) = 1.427 \times 10^{-4}$, respectively, which are slightly better than the results in the beginning example. ◁

5 Separation Bound Continuation

In this section, we consider another way to enlarge separation bounds. The key idea is that the separation bound defines a kind of ε-neighborhood of the given polynomial $f(x,y)$. From this point of view, we consider to continuate one neighborhood to others like analytic continuations.

For the given $f(x,y)$ and $0 < b \in \mathbb{R}$, let $\mathcal{A}_b(f)$ be the set of all $\tilde{f} \in \mathbb{C}[x,y]$ with $\|f-\tilde{f}\|_2 < b$ and $\deg(\tilde{f}) \leq \deg(f)$. Hence $\mathcal{A}_{B(f)}(f)$ denotes a ε-neighborhood of the given $f(x,y)$, in which all polynomials must remain absolutely irreducible.

Definition 1. *Let $B_0(f) = B(f)$ and $B_i(f) \in \mathbb{R}$ $(i = 1, \ldots)$ be the maximum value satisfying*

$$\mathcal{A}_{B_i(f)}(f) \subseteq \bigcup_{g \in \mathcal{A}_{B_{i-1}(f)}(f)} \mathcal{A}_{B(g)}(g).$$

We call $B_i(f)$ $(i > 0)$ and $B_\infty(f)$ a continuated separation bound and the maximum continuated separation bound, of $f(x,y)$, respectively. ◁

One may think that "Does the given polynomial have an approximate factorization with tolerance $B_\infty(f)$?". The author thinks that the answer is "No" since separation bounds by the known methods are far from backward tolerances with which the given polynomials have approximate factorizations. However, this continuation helps us to enlarge separation bounds as follows.

For the problem 1, let ε be a arbitrary positive real number and $\Delta, b \in \mathbb{R}$ be

$$\Delta = B(f)/\sqrt{(n+1)(m+1)} - \varepsilon, \quad b = \min_{0 \leq i \leq n, 0 \leq j \leq m, k=-1,1} B(f + k\Delta x^i y^j).$$

For the problem 2, let ε be a arbitrary positive real number and $\bar{\Delta}, \bar{b} \in \mathbb{R}$ be

$$\bar{\Delta} = \bar{B}(f)/\sqrt{\#\mathcal{M}} - \varepsilon, \quad \bar{b} = \min_{x^i y^j \in \mathcal{M}, k=-1,1} \bar{B}(f + k\bar{\Delta} x^i y^j),$$

where $\mathcal{M}$ denotes the set of all the monomials $x^i y^j$ satisfying $\mathcal{P}(x^i y^j) \subseteq \mathcal{P}(f)$.

Lemma 5. *$\sqrt{b^2+\Delta^2}$ and $\sqrt{\bar{b}^2+\bar{\Delta}^2}$ are also separation bounds $B(f)$ and $\bar{B}(f)$ of $f(x,y)$, respectively, and they may be better than the original bounds.* $\triangleleft$

Proof. We give the following proof only for $\sqrt{b^2+\Delta^2}$ since that for $\sqrt{\bar{b}^2+\bar{\Delta}^2}$ is proved by the same way. Let a polynomial $\tilde{f} \in \mathcal{A}_{\sqrt{b^2+\Delta^2}}(f)$ be

$$\tilde{f} = \sum_{i,j}(c_{i,j} + \tilde{c}_{i,j})x^i y^j.$$

By the definition of $B(f)$, we have that $\tilde{f}$ is absolutely irreducible if $|\tilde{c}_{i,j}| \leq \Delta$ for all i and j. Hence, we suppose that one of variations of coefficients of $\tilde{f}$ from f is larger than Δ and such the term be $x^{i'}y^{j'}$. We rewrite $\tilde{f}$ be

$$\tilde{f} = \sum_{i,j}(c_{i,j} + \acute{c}_{i,j})x^i y^j + k\Delta x^{i'}y^{j'}, \ \ (k = -1 \text{ or } 1).$$

We have $\| f - \tilde{f} \|_2^2 = \sum_{i,j}|\acute{c}_{i,j}|^2 + 2|\acute{c}_{i',j'}|\Delta + \Delta^2 < b^2 + \Delta^2$ which means $\sum_{i,j}|\acute{c}_{i,j}|^2 < b^2$. Therefore, we have $\tilde{f} \in \mathcal{A}_b(f + k\Delta x^{i'}y^{j'})$ meaning $\tilde{f}$ remains absolutely irreducible, and the lemma is valid. $\square$

Using the lemma, we define partial continuated separation bounds of $f(x,y)$, $B_C(f) = \max\{B(f), \sqrt{b^2+\Delta^2}\}$ and $\bar{B}_C(f) = \max\{\bar{B}(f), \sqrt{\bar{b}^2+\bar{\Delta}^2}\}$.

Example 3. For the polynomial in the example 1, the algorithm using the above lemma (let it be Algorithm C) outputs $B_C(f) = 4.068 \times 10^{-5}$ and $\bar{B}_C(f) = 1.467 \times 10^{-4}$, which are slightly better though it is very time-consuming. $\triangleleft$

6 Numerical Experiment and Remarks

We have generated 100 bivariate sparse polynomials of degrees 6 and 5 w.r.t. x and y, respectively, with coefficients randomly chosen in the real interval $[-1, 1]$, where each sample is irreducible and about 25% of coefficients are non-zero. With those polynomials, we have tested the new algorithm 1, 2 and C, using our preliminary implementations. We note that the results of our experiments are small so we have to take care of precisions. Basically, we have tested it using the same way in the paper [3] (bounding errors of singular values). The upper part of the table 1 shows the results. According to the results, our improvements give us more accurate separation bounds.

Moreover, we have generated 100 bivariate reducible polynomials. Each polynomial is a product of two dense polynomials of total-degrees 5 and 4, respectively, with coefficients randomly chosen in the integer interval $[-5, 5]$. Using those polynomials, we have generated 100 approximately reducible polynomials. Each polynomial is a sum of a reducible polynomial and a polynomial which has the same degree as the reducible polynomial, about 25% as many terms and coefficients randomly chosen in the real interval $[-10^{-4}, 10^{-4}]$.

With those polynomials, we have tested the new algorithms except for the algorithm C . The lower part of the table 1 shows the results. According to the

Table 1. Experimental results

		Algorithm	$B(f)/\|\|f\|\|$ or $\bar{B}(f)/\|\|f\|\|$	Ratio to KM03's
Irr.	$B(f)$	KM03	1.412×10^{-2}	–
		Algorithm 2 in [1]	1.463×10^{-2}	1.036
		Algorithm C	1.473×10^{-2}	1.043
	$\bar{B}(f)$	KM03 (Polytope)	1.639×10^{-2}	–
		Algorithm 1	1.661×10^{-2}	1.013
		Algorithm 2	1.680×10^{-2}	1.024
		Algorithm C	1.703×10^{-2}	1.038
Red.	$B(f)$	KM03	1.074×10^{-6}	–
		Algorithm 2 in [1]	1.083×10^{-6}	1.008
	$\bar{B}(f)$	KM03 (Polytope)	2.145×10^{-6}	–
		Algorithm 1	2.177×10^{-6}	1.015
		Algorithm 2	2.204×10^{-6}	1.027

results, our improvements give us more accurate separation bounds. Although we could not use the algorithm C for all the generated polynomials due to its time-complexity, it gave us better results. We note that an average of backward errors of those approximately reducible polynomials by the method [9] is 2.829×10^{-4}.

The methods revised by the former and this, are more time-consuming than the originals though their separation bounds are better. The reason is that we have to compute singular values after deleting unnecessary rows. Furthermore, the author wishes to thank the anonymous referees for their suggestions.

References

1. Nagasaka, K.: Towards more accurate separation bounds of empirical polynomials. SIGSAM/CCA **38** (2004) 119–129
2. Kaltofen, E.: Effective noether irreducibility forms and applications. J. Computer and System Sciences **50** (1995) 274–295
3. Nagasaka, K.: Towards certified irreducibility testing of bivariate approximate polynomials. In: Proc. ISSAC '02. (2002) 192–199
4. Sasaki, T.: Approximate multivariate polynomial factorization based on zero-sum relations. In: Proc. ISSAC 2001. (2001) 284–291
5. Nagasaka, K.: Neighborhood irreducibility testing of multivariate polynomials. In: Proc. CASC 2003. (2003) 283–292
6. Kaltofen, E., May, J.: On approximate irreducibility of polynomials in several variables. In: Proc. ISSAC '03. (2003) 161–168
7. Ruppert, W.M.: Reducibility of polynomials $f(x, y)$ modulo p. J. Number Theory **77** (1999) 62–70
8. Gao, S., Rodrigues, V.M.: Irreducibility of polynomials modulo p via newton polytopes. J. Number Theory **101** (2003) 32–47
9. Gao, S., Kaltofen, E., May, J., Yang, Z., Zhi, L.: Approximate factorization of multivariate polynomials via differential equations. In: Proc. ISSAC '04. (2004) 167–174

Compiler-Enforced Memory Semantics in the SACLIB Computer Algebra Library

David G. Richardson and Werner Krandick

Drexel University, Philadelphia PA 19104, U.S.A.
{richardson, krandick}@cs.drexel.edu

Abstract. We present a memory management subsystem for the computer algebra library SACLIB that removes the potential for memory leaks or double deletes in applications using the system. The system encapsulates the management of arrays that are allocated on the heap or on the system stack. The system makes arrays responsible for their own memory management, and enables the compiler to prevent other parts of SACLIB from managing array memory. To prove that our memory module and all applications using it are leak free and double delete free we introduce a new iterator concept and implement a model of that concept using generic programming techniques such as template metaprogramming. Our innovations reduce the amount of code responsible for array memory management from 10,000 lines of code to 2,000 lines of code. Using hardware performance counters we show optimizing compilers are capable of avoiding any runtime overhead.

1 Introduction

Memory management is a critical component of any computer algebra library. Correct memory management is required to ensure the correctness of subroutines; efficient memory management facilitates improved execution speed and the processing of larger data sets.

The SACLIB [1,2] library of computer algebra programs is used as the basis of the quantifier elimination systems QEPCAD [3,4,5] and QEPCAD B [6,7]. However, SACLIB contains a number of memory management defects, all of which occur outside of SACLIB's garbage collector. Memory leaks involving dynamically allocated arrays have caused problems in certain large computations [8]. While runtime-tools such as Valgrind [9], Electric Fence [10], and Purify [11] can be used to detect some memory defects, the tools cannot guarantee their absence. This is unfortunate because dynamic arrays are used extensively in a weakly typed computer algebra system such as SACLIB.

We port SACLIB from C to C++ and introduce a memory management subsystem that removes the potential for any memory leaks or double deletes in applications using the system. The system encapsulates the management of arrays that are allocated on the heap or on the system stack. The system makes arrays responsible for their own memory management, and enables the compiler to prevent other parts of SACLIB from managing array memory. To prove that our memory module and all applications using it are leak free and double delete free we introduce a new iterator concept (see Section 3.3) and implement a model of that concept using generic programming techniques such as template meta-programming. Our innovations reduce the amount of code responsible for array memory management from $10,000$ lines of code to $2,000$ lines of code.

Our memory management system consists of an iterator to control memory usage and of array classes to handle memory allocation. Using template meta-programming [12,13,14] we implement a model of our iterator concept that handles iterators of arbitrary dimension. Our meta-programming techniques are similar to those used by Musser and Schupp in the SuchThat [15,16,17] computer algebra library and in library-assisted code optimizations [18,19]; Schreiner [20] details three other computer algebra libraries that use generic programming. However, our implementation is the first to provide iterators capable of protecting memory of arbitrary dimension.

The array component of our memory management system (Sections 3.1 and 3.2) is similar to the `std::vector` class found in the Standard Template Library [21,22,23], but our implementation is faster than the `std::vector` class of the implementations shipped with our compilers, see Table 1.

In addition, our memory management subsystem offers numerous software maintenance advantages. It allows some runtime bounds checking to be performed during the development process without the need for external tools. The system provides a well-defined architecture for future extensions of the non-garbage collected memory subsystem. By embedding memory semantics directly in the C++ type system, the assumptions functions make about memory management are clearly announced and enforced by their prototypes.

To attain these benefits, it was necessary to port SACLIB from C to C++. We implemented code transformation scripts to convert most of the code from K&R C [24] to ANSI C++ [21,22]; the remainder of the code was related to floating point exception handling and was converted by hand.

In Section 2 we detail the limitations of the design of SACLIB that resulted in a strong potential for memory leaks. In Section 3 we present our memory management subsystem, outline the general implementation techniques employed, and give examples of its use. In Section 3.4 we explain why our memory management subsystem and the applications using it are leak free and double delete free. In Section 4 we report the results of performance experiments on our memory management subsystem. Through the use of hardware performance counters and the PAPI library [25] we are able to report timings with a resolution under 50 nanoseconds. In Section 5 we explain how we ported SACLIB from C to C++. We conclude in Section 6 with a discussion of the effectiveness of our method and of the techniques that make our implementation possible.

2 Existing SACLIB Practice

2.1 The Source-Sink Idiom and Its Limitations

SACLIB uses the source-sink idiom [26] for all memory management outside of its garbage collector and some performance critical sections. In the source-sink idiom a *source function* is used to request a handle to a system managed resource. The handle allows clients of the source function to access the resource without any knowledge of how it was acquired by the source function. When the handle provided by the source function is no longer needed, it is passed to a *sink function* which releases the resource referred to by the handle. A resource leak occurs when all handles to a resource have been lost, and, as a result, the resource cannot be released by a sink function. Examples of source-sink pairs are the library functions `malloc`/`free` and SACLIB functions such as `GETARRAY`/`FREEARRAY` and `GETMATRIX`/`FREEMATRIX`.

The source-sink idiom gives rise to two types of programming mistakes, and it makes it hard to change the implementation of the source-sink pair.

Failure to Release Resources. The source-sink idiom cannot guarantee that all resources acquired from a source function will be released by a sink function. Indeed, memory leaks in SACLIB subroutines prevented an experiment involving a large number of polynomials from successfully running to completion [8]; Figure 1 shows an example of a SACLIB routine with a memory leak.

```
Step3: /* Compute a list of the order 2 minors. */
        C = GETARRAY(2);
        C[0] = 0;
        C[1] = 1;
        L = NIL;
        while (C[0] >= 0) {
           i = C[0];
           j = C[1];
           B1 = IPPROD(r,A[n-2][i],A[n-1][j]);
           B2 = IPPROD(r,A[n-2][j],A[n-1][i]);
           B = IPDIF(r,B1,B2);
           c = CSFAM(2,C);
           L = COMP2(B,c,L);
           LEXNEXTC(n,2,C,&t); }
        L = INV(L);
        /* FREEARRAY(C) missing. */
```

Fig. 1. Incorrect use of the source-sink idiom in the SACLIB routine MAIPDME results in a memory leak

Uninitialized Delete and Double Delete. The source-sink idiom cannot guarantee that the source and sink functions are called for valid handles. When the same handle is passed to a sink function more than once it is called *double deletion*. When a handle not initialized by a source function is passed to a sink function this is called *uninitialized deletion*. In SACLIB, uninitialized deletions produce undefined runtime behavior.

Source-Sink Functions Difficult to Maintain. SACLIB uses pointers for all of its handles. Because the compiler has no control over what functions are called on these pointers, clients of the source-sink pairs must check the implementation of the source-sink pairs to determine what operations may safely be applied to the handles. Not surprisingly, this poor information hiding has all of the problems one would expect [27]. In addition to making current development more difficult by requiring the programmer to be aware of the implementation of the source-sink pairs, changes to the source-sink pairs can cause other previously working parts of SACLIB to become defective.

2.2 Stack-Before-Heap Idiom

For some performance critical sections of SACLIB the stack-before-heap idiom is used for memory management. The idiom consists of allocating memory on the stack that is large enough for the majority of inputs that SACLIB is expected to run on. In the event that the amount of stack memory is sufficient for a given input, the overhead of both allocating heap memory and heap fragmentation are avoided. This technique is similar to the small string optimization [28,29], although SACLIB does not use unions.

The stack-before-heap idiom is more error prone than the source-sink idiom. It is more complicated and does not lend itself to being encapsulated in functions. As a result, all uses of the stack-before-heap idiom require the programmer to manually implement all parts of the idiom. Even when implemented correctly, this detracts from the readability of the code.

Even when the stack-before-heap idiom is implemented correctly, it is not always a performance optimization. It is only a performance optimization when the amount of stack memory is sufficient for most inputs. If inputs frequently (or even worse, always) require more memory than is allocated on the stack, the stack-before-heap idiom offers the opportunity for reduced performance. Furthermore, because the stack before heap idiom is manually implemented, each occurrence must be manually checked to see if the idiom is in fact an optimization.

While it is likely that the stack before heap idiom originally was an optimization for a meaningful input set, it is unlikely that the use of the idiom will be rechecked as system maintenance is performed and the program is used with new input sets. There is no way to automatically detect where all uses of the stack-before-heap idiom have been used. In even a moderately sized system such as SACLIB, it becomes infeasible to test each occurrence of the stack-before-heap idiom as the system evolves.

2.3 Memory Passing and Return Value Conventions

All memory in SACLIB is referred to by pointers. Functions requiring the ability to be called polymorphically with respect to the value type of a pointer are implemented using void pointers. Neither pointers or the data they point to are declared const.

SACLIB currently uses pointers to return values from functions that return more than one value. By convention, the output parameters of a function are placed at the end of the argument list and the argument names have a trailing underscore appended.

Pointers are one of the most powerful, flexible—and error prone—features of the C and C++ programming language. Because of their flexibility, it is generally the case that a function taking pointers as its arguments does not require all of the functionality offered by pointers. This results in function prototypes that fail to convey as much information about the function as more strongly typed prototype can. This lack of information results in three types of maintenance problems.

1. A function taking raw pointers does not provide any information about its expectations regarding memory ownership.
2. Pointers intended to be used solely as return values are capable of being used for pointer arithmetic.
3. Functions may have memory defects added during maintenance, and the function prototype cannot convey any memory ownership information.

3 Improvements to SACLIB

3.1 Replacing the Source-Sink Idiom with RAII

A well know replacement for the source sink-idiom is the Resource-Acquisition-Is-Initialization (RAII) idiom [12,30,26,31,32]. This idiom is used extensively in modern C++ libraries, for example containers in the Standard Template Library [33], the Loki smart pointer classes [12], and the Boost smart pointer library [34].

In the RAII idiom, responsibility for resource acquisition is delegated to a class specifically designed to safely acquire and release resources. The class is responsible for acquiring resources in its constructor and freeing all resources it has acquired in its destructor. The C++ standard [21,22] guarantees that an object will have its destructor called when it goes out of scope.

In SACLIB we replaced the GETARRAY/FREEARRAY source-sink pair with an `array` class. To show how the RAII idiom can be implemented we give a minimal implementation of the array class in Figure 2 (Left). The actual implementation contains additional features such as the performance enhancements detailed in Section 3.2, the typedefs and methods required to make the array model the STL Random Access Container concept, and the ability to resize the array after the constructor is called. Figure 2 (Right) shows that the use of our `array` class automatically removes memory leaks from SACLIB.

```
template<typename T>
class array{
    public:
       //ctors acquire memory
          array():activeArray_(NULL){};
          array(size_t n):activeArray_(new T[n]){};
       //dtor releases memory
          ~array(){ delete[] activeArray_; }

       //allow the array class to be used like a built-in array
          T& operator[](ptrdiff_t index){
             return activeArray_[index];
          };//operator[]
          const T& operator[](ptrdiff_t index) const {
             return activeArray_[index];
          };//operator[] const
   private:
       T *activeArray_;
};//array
```

```
Step3: /* Compute a list of the order 2 minors. */
          array<BDigit> C(2);
          C[0] = 0;
          C[1] = 1;
          L = NIL;
          while (C[0] >= 0) {
             i = C[0];
             j = C[1];
             B1 = IPPROD(r,A[n-2][i],A[n-1][j]);
             B2 = IPPROD(r,A[n-2][j],A[n-1][i]);
             B = IPDIF(r,B1,B2);
             c = CSFAM(2,&C[0]);
             L = COMP2(B,c,L);
             LEXNEXTC(n,2,&C[0],&t); }
          L = INV(L);
```

Fig. 2. Left: The heap **array** class cannot leak memory. Right: The new version of the SACLIB routine MAIPDME is leak-free—even without the call to FREEARRAY that was missing in Figure 1.

Remark 1. *An* `array2d` *class is also provided for the new SACLIB. It provides RAII memory management for two-dimensional arrays and offers an interface similar to the* `array` *class.*

3.2 Encapsulation of the Stack-Before-Heap Idiom

The principal drawback to the stack-before-heap idiom is the inability to encapsulate the idiom in C functions. In addition to removing the need for the source-sink idiom, the `array` class also provides a convenient place to encapsulate the stack-before-heap idiom. Figure 3 (Left) is an example of an implementation of the array class that provides an encapsulated version of the stack-before-heap idiom. Note that the implementation actually used in SACLIB contains additional features such as the typedefs and methods required to make the array model the STL Random Access Container concept, and the ability to resize the array after the constructor is called. Partial specialization is used to ensure there is no overhead when `N` is zero. Figure 3 (Right) shows an `array` used to provide a safe alternative to the stack-before-heap idiom.

```
template<typename T, size_t N>
class array{
    public:
       //ctors acquire memory
          array():activeArray_(stackStorage){};
          array(size_t n):
             activeArray_( n <= N ? stackStorage : new T[n] )
          {};
       //dtor releases memory if needed
          ~array(){
             if(activeArray != stackStorage) delete[] activeArray_;
          }//~array

       //allow the array class to be used like a built-in array
          T& operator[](ptrdiff_t index){
             return activeArray_[index];
          };//operator[]
          const T& operator[](ptrdiff_t index) const {
             return activeArray_[index];
          };//operator[] const
   private:
       T *activeArray_;
       T stackStorage_[N];
};//array
```

```
void encapsulated_stack_before_heap_example(){
    //how much stack memory to use
       const size_t stack_size(50);
    //''compute'' the size of array
       int n;
       cout << ``What size array do you want?'';
       cin >> n;
    //acquire resources with a constructor using
    //the stack-before-heap idiom
       array<int,stack_size> a(n);//pass desired
                                  //array size to
                                  //constructor
}//encapsulated_stack_before_heap_example
```

Fig. 3. Left: The stack `array` class efficiently stores elements on the system stack. Right: Memory leaks are automatically avoided and the system stack is used for storage using the stack `array` class.

The `array` class has been given an additional template parameter `N` to specify the number of elements to be stored on the stack and the private member array `stackStorage_` in which to store up to `N` elements of type `T`. In the event that a user instantiates an array class requiring less than `N` elements, the array constructor will set the `activeArray_` member to `stackStorage_` and there will be no need to allocate memory from the heap.

While the removal of memory leaks is the most substantial benefit obtained of this implementation, there is a another benefit. In the manual implementation of the stack-before-heap idiom it is difficult to verify that the amount of storage allocated on the stack is sufficient to be a performance optimization because there is no central location to check. With the encapsulation of the idiom in the array class a check can be placed in the array constructor. With conditional compilation all overhead of the check can be avoided in production code. By altering only methods of the array class, all clients of the array class can be run on representative inputs to see if the proper amount of stack storage space is being allocated. Note: `alloca` was not considered because it would have complicated the porting of SACLIB from C to C++.

3.3 Recursively Fixed Iterator

The heavy use of the source-sink idiom in SACLIB offered the opportunity to allow functions to express and enforce memory ownership expectations explicitly through their prototypes. The key insight required to allow memory ownership to be expressed by the function prototype is this: non-sink functions taking pointers to memory managed by the source-sink idiom should not deallocate the memory referred to by the pointers or overwrite the pointers.

Figure 4 shows an example of pointer usage in SACLIB. P is a two-dimensional pointer of type int**. The dimensionality of a C++ type is defined as the number of times * can legally be applied to it. Variables with a dimensionality greater than 0 (pointers) form the structure of the memory, while variables with dimensionality of 0 (not pointers) form the contents of the memory. Functions that manipulate the structure of a pointer are called memory managers of the pointer, while functions that do not manipulate the structure of a pointer are called contents-only users of the pointer.

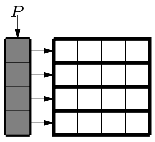


Fig. 4. P is a pointer to a SACLIB matrix. The gray memory cells contain pointers and are the structure of P. The white memory cells contain integers and are the contents of P.

To facilitate the protection of memory structure, we define the following concept.

Definition 1. *Let T be a C++ type. T is a model of the* recursively fixed iterator *concept if*

1. T provides "T::operator*" and "T::operator[]" with pointer semantics,
2. T provides "T::operator-¿" with the semantics "(T::operator*())."",
3. T provides all pointer arithmetic operators that are not self-modifying (+,-, ¡, ¡=, ¿, ¿=, ==), and
4. For all instances t of type T functions that do not use `const_cast` may write to the contents of t, but if they write to the structure they will not compile.

Implementation. Using operator overloading and template meta-programming we have implemented the `rec_fixed_itr` class as a model of the recursively fixed iterator concept.

Using the operator overloading features of C++ it is straightforward to provide operator*, operator-¿, operator[], and the non-mutating pointer arithmetic operators for `rec_fixed_itr`. Such implementation techniques are common practice and have been well documented in libraries such as Loki [12] and Boost [34]. The C++ const keyword allows variables to be made read-only.

The difficulty in implementing rec_fixed_itr occurs in deciding the return types to give to operator*, operator-¿, and operator[]. When they are providing access to memory that is in the structure of a `rec_fixed_itr` instance their return type must prevent writes and deletes. When they are providing access to the contents they must be mutable. For this, we must be able to determine at compile time if a memory de-reference will access the contents or the structure of a `rec_fixed_itr` instance.

We use C++ template meta-programming to accomplish this. Originally, the C++ template instantiation mechanism was intended to provide a type-safe alternative to macros for the generation of families of functions that differ only in their types. But then Erwin Unruh [14] discovered that the C++ template instantiation mechanism offered both branching and recursion, allowing arbitrary computation at compile time. The C++ template instantiation mechanism offers a Turing-complete functional language that executes at compile time.

The presence of a Turing-complete programming language in the C++ compilation process has been widely used in libraries such as Boost [34], Loki [12], Blitz++ [35,36], and GMCL [36,37] to provide both optimizations and improved client interfaces. There are several excellent books [13,12,37] on library design using template meta-programming in C++.

Definition 2. *A C++ template meta-function is a C++ class template that takes its parameters as template arguments and returns types or integral constants as nested members.*

The key to determining if a memory de-reference will return an element of the structure lies in determining the dimension of the variable returned by the de-reference. Since the dimension of a variable is determined by its type, we can reduce the problem to computing the dimension of an arbitrary type. Figure 5 (Left) shows an implementation of the template meta-function `Dim` we use to compute the dimension of a type. The function is defined recursively. The dimension of the `NullType` is defined as zero and used as the base case of the recursion. The `NullType` is a placeholder type used to represent an unusable type, and serves a similar role as `null` does for pointers. If the argument to the `pointsTo` meta-function is a C++ pointer, it returns the type pointed to by the argument; otherwise it returns the `NullType`. The `isNullType` meta-function returns `true` if its argument is the `NullType`; otherwise it returns `false`. The `if_value` meta-function returns its second argument if its first argument is `true`; otherwise it returns its third argument. We implemented all meta-functions called by `Dim` for the sole purpose of computing the return types of the pointer operators of the rec_fixed_itr concept. In total, 22 meta-functions were required for the computation of all needed return types.

```
//Meta-function:  Dim
//Arguments:      T - a type
//Description:    Computes the dimensionality of T.
//Returns:        value: the dimension of T
      template<typename T> struct Dim;
      template<> struct Dim<NullType>{
         enum { value = 0 };
      };//struct Dim<NullType>
      template<typename T>
      struct Dim{
         enum {
            value = if_value<
                       isNullType<typename pointsTo<T>::type>::value,
                       Dim<NullType>::value,
                       1 + Dim< typename  pointsTo<T>::type>::value
                    >::value
         };//enum

      };//struct
```

```
template<
   typename     CType,
   unsigned int Dimension = Dim<CType>::value
>
class rec_fixed_itr{
   public:
      typedef rec_fixed_itr<CType,Dimension> self;
      //ctor
         rec_fixed_itr(const CType proxy):
            proxy_(proxy)
         {};//rec_fixed_itr

      //example of return type computation
         typedef
            typename return_type<self>::op_star_type
            op_star_ret
         ;
         inline op_star_ret operator*(){
            return op_star_ret(*proxy_);
         };//operator*
   private:
      CType proxy_;
};
```

Fig. 5. Left: The meta-function Dim computes the dimensionality of a type. Right: Outline of the rec_fix_itr implementation technique.

Figure 5 (Right) offers an outline of the implementation of the rec_fixed_itr. There are four major techniques to take note of. (1) The constructor is declared without the keyword explicit. This allows the compiler to construct rec_fixed_itrs implicitly from variables of the CType. As a result, functions declared to take an argument of type rec_fixed_itr<T*> may still be passed variables of type T*. (2) The dimension of the CType is computed at template instantiation and embedded in the rec_fixed_itr type. This allows the dimension to be passed, via the typedef self, to the meta-function return_type for computation of the return type. (3) The proxy_ member is always wrapped in an object of a type computed by the return_type meta-function before being returned to clients. This allows the proxy_ to be used in the explicit construction of other types. The meta-function return_type returns as types rec_fixed_itr with a lower dimension than the current CType, const references, and mutable references. This construction of return values allows the return_type to compute types that will protect the structure of the memory accessible from proxy_ while at the same time allowing access to the contents.

Note: The actual rec_fixed_itr implementation is significantly more complicated. All told, the rec_fixed_itr and its 22 supporting meta-functions comprise 666 lines of code. The full implementation handles all pointer arithmetic operations required to support the full range of pointer functionality specified by the recursively fixed iterator concept and validates that rec_fixed_itr is instantiated correctly.

With the availability of the rec_fixed_itr class functions can become both more expressive and capable of protecting memory structure passed to them. Consider:

```
void might_be_a_memory_manger_of_p(int ***p);
void contents_only_user_of_p(rec_fixed_itr<int***> p);
```

From the function prototype, neither the compiler nor maintenance programmers have any knowledge of the usage patterns the developers of `might_be_a_memory_manager_of_p` intended. However, the function prototype of `contents_only_user_of_p` clearly states it is a contents only user via the argument type `rec_fixed_itr<int***>`. If the definition of `contents_only_user_of_p` ever attempts to be a memory manager of `p`, the compiler will prevent compilation of `contents_only_user_of_p`. Furthermore, we implemented `rec_fixed_itr` such that the compiler will emit a diagnostic error that indicates the line of `contents_only_user_of_p` that attempts to manage memory. Schreiner's criticism regarding template parameter validation [20] no longer applies; indeed, the static assert facilities of Boost [34] allowed us to provide meaningful error messages and parameter validation.

3.4 Correctness of the Memory System

Resource leaks can only occur when all handles to a resource are lost. The last handles to a resource can be lost in two ways: (1) the variable holding the handle goes out of scope or (2) the variable holding the handle is assigned a new handle.

By encapsulating memory handles in the array class, we guarantee that the destructor of the array will release memory when instances of the array class go out of scope. This removes the possibility for the memory leaks via situation (1).

By passing memory handles into functions via the `rec_fixed_itr`, it becomes impossible to write to the memory structure referred to by the `rec_fixed_itr`. As the `rec_fixed_itr`'s structure contains all of the memory handles accessible from the `rec_fixed_itr`, it is not possible to overwrite—and thus loose—a memory handle.

Double deletes can only occur when a memory handle is deleted twice. The array class will delete memory only once in its destructor. Since the `rec_fixed_itr` does not allow memory to be deleted at all, a double delete cannot occur.

4 Performance Testing

In order to obtain the most accurate timing possible we used hardware performance counters to measure the number of CPU cycles required for different methods of memory management. All measurements were obtained on a 3.0 GHz Pentium 4 (1024KB L2 cache) running Fedora Core 2. The Performance Counter API (PAPI) [25] version 3.0.7 was used to collect all measurements. The single processor 2.6.5 kernel shipped with Fedora Core 2 was patched with the perfctr-2.6.x patch distributed with PAPI. Tests were compiled with g++ version 3.3.3 [38] and icc [39] version 8.0. Both compilers were given the flags *-O3 -march=pentium4 -DNDEBUG*.

Experiments were repeated 500 times for each compiler. Cycle counts were obtained with `PAPI_get_virt_cyc`. Timings were discarded if they were more than twice the mean. Thirty-one measurements were discarded for gcc; 40 were discarded for icc. Average times are reported in Table 1.

Due to our implementation, there can only be run-time overhead in our classes with respect to raw pointers if the optimizer fails to inline and collapse references down to machine code equivalent to the raw pointer case. The icc results show that compilers can avoid this abstraction penalty from our class interface; the gcc results show that not all compilers do. Manual inspection of the machine code generated by gcc confirms the performance overhead is from an inability to properly collapse references. Both results show that our specialized classes perform better than the general-purpose `std::vector`.

Table 1. Machine cycles for various types of memory management. *Pointer* designates the non-garbage collected memory management used by the old SACLIB, *array* stands for our management of heap-allocated memory, *array_stack* for our management of memory that is allocated (more efficiently) on the system stack, and *vector* refers to the vector-class available in STL. The table reports the average number of machine cycles required for allocating and deallocating 500 elements, and for reading and writing 500 elements. For *pointer* alloc denotes `new` and dealloc denotes `delete[]`. For *array*, *array_stack*, *array2d*, and *vector* alloc denotes their constructor and dealloc their destructor. In all cases, read denotes an expression of the form a = p[i] and write denotes an expression of the form p[i] = a.

icc 8.0 benchmarks

1-dim	alloc	read	write	dealloc
pointer	9357.3	1554.2	2785.6	5734.1
array	9359.7	1561.6	2786.8	5782.2
array_stack	931.2	1564.9	2751.3	223.6
vector	10867.1	1521.2	2776.9	6964.7

2-dim	alloc	read	write	dealloc
pointer2d	25046.8	2377.5	5108.0	13022.8
array2d	21951.6	1998.9	5193.9	13409.8
vector2d	35301.1	3524.0	6314.1	20654.3

gcc 3.3.3 benchmarks

1-dim	alloc	read	write	dealloc
pointer	7081.1	6317.5	7249.7	4581.6
array	7079.6	7294.0	7252.3	4679.5
array_stack	986.0	7340.0	7251.5	203.1
vector	10279.0	7315.1	7269.3	5863.0

2-dim	alloc	read	write	dealloc
pointer2d	17644.2	7719.1	7741.9	12023.0
array2d	20466.1	7856.7	9248.5	12911.0
vector2d	31682.2	8073.4	11844.6	17039.2

5 Automatic Conversion

The old SACLIB was written in K&R C [24]. In K&R C functions can be defined as

```
void function(a,b)
   int a;
   double b;
{

}
```

While this is a legal function definition in both K&R C and ANSI C [40,41], it is not a legal function definition in ANSI C++ [21,22]. In ANSI C++, the function must be defined as

```
void function(int a, double b){

}
```

Additionally, K&R C requires only the return type of a function to be known prior to allowing it to be called. The above function would have its return type described as

```
void function();
```

ANSI C allows a full function prototype that specifies both the return type of the function and the types of the arguments

```
void function(int a, double b);
```

ANSI C++ requires the use of a full function prototype.

SACLIB provided full function prototypes for compilers that were ANSI C compliant. The function prototypes were wrapped in a macro

```
void function P__((int a, double b));
```

where P__ was defined as

```
#ifdef __STDC__
   #define P__(A) A
#else
   #define P__(A) ()

#endif
```

This ensured that function prototypes were only used by ANSI C compilers. We first compiled SACLIB using an ANSI C compiler to ensure the correctness of the prototypes. Once we knew the prototypes matched, a perl script was written that read the existing prototypes, took the arguments from the prototypes and placed them in the function definitions, and deleted the old K&R function argument definitions. Another perl script was then used to remove the P__ macros.

As SACLIB was a generally well-structured C program, this was all that was required to port SACLIB from C to C++.

6 Conclusion

We have shown how several features of the C++ language can be combined with our recursively fixed iterator concept and the RAII idiom to ensure correct allocation and deallocation of memory. User extensibility of the C++ type system, operator overloading, and template meta-programming allow the compiler to verify the absence of memory defects.

As a result, SACLIB and any future extensions using our memory subsystem will be leak free and double delete free. We have significantly improved the self documenting nature of the SACLIB code base and substantially reduced the likelihood of memory errors. Consequently, programmers will spend less time detecting, diagnosing, and repairing memory defects.

Our experiments show that these benefits can be obtained without degradation of performance.

References

1. Collins, G.E., et al.: SACLIB User's Guide. Technical Report 93-19, Research Institute for Symbolic Computation, RISC-Linz, Johannes Kepler University, A-4040 Linz, Austria (1993)
2. Hong, H., Neubacher, A., Schreiner, W.: The design of the SACLIB/PACLIB kernels. Journal of Symbolic Computation **19** (1995) 111–132
3. Collins, G.E., Hong, H.: Partial CAD contruction in quantifier elimination. Technical Report OSU-CISRC-10/89 TR45, Computer Science Department, Ohio State University (1989)
4. Hong, H.: An improvement of the projection operator in cylindrical algebraic decomposition. In: Proceedings of the International Symposium on Symbolic and Algebraic Computation, ACM Press (1990) 261–264
5. Hong, H., Liska, R., Steinberg, S.: Testing stability by quantifier elimination. Journal of Symbolic Computation **24** (1997) 161–187
6. Brown, C.W.: QEPCAD B: A program for computing with semi-algebraic sets using CADs. SIGSAM Bulletin **37** (2003) 97–108
7. Brown, C.W.: QEPCAD B: A system for computing with semi-algebraic sets via cylindrical algebraic decomposition. SIGSAM Bulletin **38** (2004) 23–24
8. Krandick, W., Mehlhorn, K.: New bounds for the Descartes method. Journal of Symbolic Computation (to appear)
9. Valgrind Developers: Valgrind. (`http://valgrind.kde.org`)
10. Electric Fence Developers: Electric Fence. (`http://www.pf-lug.de/projekte/haya/efence.php`)
11. IBM Corporation: Rational Purify. (`http://www-306.ibm.com/software/awdtools/purify/`)
12. Alexandrescu, A.: Modern C++ Design. Addison Wesley Longman, Inc. (2001)
13. Abrahams, D., Gurtovoy, A.: C++ Template Metaprogramming. Addison Wesley Longman, Inc. (2005)
14. Vandevoorde, D., Josuttis, N.M.: C++ Templates. Addison Wesley Longman, Inc. (2003)
15. Musser, D., Schupp, S., Loos, R.: Requirement oriented programming concepts, implications, and algorithms. In [42]. 12–24
16. Musser, D.R., Shao, Z.: Concept use or concept refinement: An important distinction in building generic specifications. In George, C., Miao, H., eds.: Formal Methods and Software Engineering: 4th International Conference on Formal Engineering Methods, ICFEM 2002 Shanghai, China, October 21-25, 2002. Proceedings. Volume 2495 of Lecture Notes in Computer Science, Springer-Verlag (2002) 132 – 143
17. Schupp, S., Loos, R.: SuchThat—Generic programming works. In [42]. 133–145
18. Gregor, D., Schupp, S., Musser, D.: Design patterns for library optimizations. Scientific Programming **11** (2003) 309–320
19. Schupp, S., Gregor, D., Musser, D., Liu, S.: Semantic and behavioral library transformations. Information and Software Technology **44** (2002) 797–810
20. Schreiner, W., Danielczyk-Landerl, W., Marin, M., Stöcher, W.: A generic programming environment for high-performance mathematical libraries. In [42]. 256–267
21. International Standards Organization `http://www.iso.org`: ISO/IEC 14882:2003: Programming languages—C++. (2003)

22. Stroustrup, B.: The C++ Standard: Incorporating Technical Corrigendum No. 1. John Wiley and Sons (2003)
23. Josuttis, N.M.: The C++ Standard Library. Addison Wesley Longman, Inc. (1999)
24. Kernighan, B.W., Ritchie, D.M.: The C Programming Language. 1st edn. Prentice Hall (1978)
25. Innovative Computing Laboratory: PAPI. (`http://icl.cs.utk.edu/PAPI`)
26. Sutter, H.: Exceptional C++. Addison Wesley Longman, Inc. (1999)
27. Parnas, D.: On the criteria to be used in decomposing systems into modules. Communications of the ACM **5** (1972) 1053–1058
28. Sutter, H.: More Exceptional C++. Addison Wesley Longman, Inc. (2001)
29. Meyers, S.: Effective STL. Addison Wesley Longman, Inc. (2001)
30. Meyers, S.: Effective C++. 2nd edn. Addison Wesley Longman, Inc. (1997)
31. Stroustrup, B.: The C++ Programming Language (Special Edition). Addison Wesley (2000)
32. Stroustrup, B.: The Design and Evolution of C++. Addison Wesley (1994)
33. Austern, M.H.: Generic Programming and the STL. Addison Wesley Longman, Inc. (1998)
34. Boost: Boost libraries and documentation. (`http://www.boost.org`)
35. Veldhuizen, T.L.: Arrays in Blitz++. In Caromel, D., Oldehoeft, R., Tholburn, M., eds.: Computing in Object-Oriented Parallel Environments. Volume 1505 of Lecture Notes in Computer Science, Springer-Verlag (1998) 223–230
36. Czarnecki, K., Eisenecker, U., Glück, R., Vandevoorde, D., Veldhuizen, T.: Generative programming and active libraries. In [42]. 25–39
37. Czarnecki, K., Eisenecker, U.: Generative Programming: Methods, Tools, and Applications. Addison-Wesley Professional (2000)
38. GNU: The GNU Compiler Collection. (`http://gcc.gnu.org/`)
39. Intel Corporation: The Intel C++ Compiler. (`http://www.intel.com/software/products/compilers/clin/`)
40. International Standards Organization `http://www.iso.org`: ISO/IEC 9899:1999: Programming languages—C. (1999)
41. British, S.I.: The C Standard: Incorporating Technical Corrigendum 1. John Wiley and Sons (2002)
42. Jazayeri, M., Loos, R.G.K., Musser, D.R., eds.: Generic Programming: International Seminar on Generic Programming. Volume 1766 of Lecture Notes in Computer Science. Springer-Verlag (2000)

Meta-Petro: An Expert System for Training Undergraduates in Metamorphic Rocks Recognition and Classification Using Photomicrographies

E. Roanes-Lozano[1], R. García[2], E. Roanes-Macías[1], A. Aparicio[2], and L.M. Laita[3]

[1] Universidad Complutense de Madrid, Depto. de Algebra, c/ Rector Royo Villanova s/n, 28040-Madrid, Spain
{eroanes, roanes}@mat.ucm.es

[2] Museo de Ciencias Naturales (Depto. de Vulcanología), Consejo Superior de Investigaciones Científicas, c/ José Gutiérrez Abascal 2, 28006-Madrid, Spain
mcny144@mncn.csic.es

[3] Universidad Politécnica de Madrid, Depto. de Inteligencia Artificial, Campus de Montegancedo, Boadilla del Monte, 28660-Madrid, Spain
laita@fi.upm.es

Abstract. Computer Algebra Systems (CASs) are convenient high-level programming languages that provide the programmer not only with symbolic capabilities and exact arithmetic, but also with different structures handling and plotting capabilities. We have used the CAS *Maple* as development tool for designing *Meta-Petro*, a system for training undergraduates in metamorphic rocks recognition and classification using photomicrographies. This expert system includes a collection of photomicrographies of thin sections of samples that are randomly presented to the user. The user can ask the system for details about: the different rocks, his guess of the solution and the right solution. Moreover, this information can be shown on the decision tree the system uses. As far as we know, in this field only "catalogs" with fixed photomicrographies have been developed so far.

Keywords: Metamorphic Petrology, Computer Algebra Systems, Expert Systems.

1 Introduction

1.1 Our Goal

We would like to provide undergraduate students with a computer tool (expert system) that allowed them to practice metamorphic rocks classifying.

One key feature of the package should be random selection of exercises from a set of (previously prepared) selected ones. This has clear advantages over packages presenting fixed examples:

– it makes the student's work almost a new one every time the system is restarted
– it provides the less gifted students with many more examples and exercises to practice
– it avoids the students extracting incorrect particular rules to solve the exercises (having one single example or very few examples of each "type of exercise" is dangerous: for instance, if we have just one set of samples of rocks, the student can remember that the only one with a "sharp corner" is the sample of marble, what is not the classifying technique we would like him to apply)

An excellent collection of samples of igneous and metamorphic rocks (i.e., plutonic, volcanic and metamorphic rocks) can be found in the "Atlas" [1]. But it just shows one or two samples of each rock.

The tool we would like to implement should also provide bookmarks and hyperlinks and the possibility of offering hints and helps.

1.2 The Computer Language Used

As said in the CFP of CASC'2005, "*The ongoing development of (Computer Algebra) systems, including their integration and adaptation to modern software environments, puts them to the forefront in scientific computing and enables the practical solution of many complex applied problems in the domains of natural sciences and engineering*".

We have chosen a Computer Algebra System (CAS) as development tool because we need to internally handle different structures to allocate the information (matrices and lists) as well as to produce plots of graphs. Despite a standard language like *C*, *Pascal* or *Java* could be used, from our experience in similar cases, like the complex simulation [2], where different structures and different plots were to be produced, choosing a scientific computing language such as a CAS is really time-saving.

Moreover, the use of CASs is not restricted to the standard educational applications of computer algebra: mathematics education and mathematical aspects of science education (see e.g. Chapter 3.6 of [3] or click on "All education PowerTools" at *Maple*'s web page [4]).

We have chosen the CAS *Maple* because it is a comprehensive and portable system that also offers the possibility to create *Maplets* (a specific kind of aplets) that can be run on a server where *Maple* is installed. For instance, we have already designed and developed a *Maplet*-based expert system for diagnosis and treatment of hypertensión [5]. The CAS *Mathematica* has recently released a similar system (*GUIKit*) [6] and it also offers a specific version for developing *JavaServer* pages (*webMathematica*).

1.3 Some Antecedents of Random Generation or Selection of Exercises

Of course there have been many experiences that use random generation of exercises in different fields. The spectrum is wide, ranging from primary education, like *Primtres* [7], to physics, like the award-winning *Microlab* [8], used in Spain in the early nineties.

Some of these applications in mathematics teaching are really smart, like the *Mathematica*-based *MathEdu* [9], that is able to recognize repeated errors of the user. The *Java*-based application *ConsMath* [10] uses sophisticated A.I. tools (agents) to develop dialog-based environments for problem solving. The latter compiles a collection of interactive problems about ordinary differential equations, that includes altering exercises by adding constraints, acceptance conditions and generation procedures.

There are also applications specifically oriented to assessment, like *Maple T.A.* [4] and even based on the word processor's capabilities [11].

1.4 About Random Selection or Generation of Examples and Exercises

We can classify random selection or generation of examples and exercises at different levels. We could distinguish:

i) the solution of the examples can be calculated by the computer. Only the "types" of problems are stored and the input data is randomly chosen from a given (prepared) universe.

   Example 1. A package for training students in operating with positive integers. A problem could be: 3+4=???. The computer calculates the solution in each case.

ii) the solution cannot be calculated, but the solution strongly depends on the particular problem. The "types" of problems, the corresponding possible data and corresponding "types of solutions" are stored. The "type" of problem and the input data are randomly chosen and combined. The solution is calculated using the adequate "type of solution" and the given data (applying the corresponding formula).

   Example 2. A package for training students in calculating volumes of 3D figures. The general "types" of problems could be like: "calculate the volume of a cube of side ..." and the input data would be numbers to fill the "..." (e.g. "1m"). The computer takes the formula to be used and calculates the solution for the given data.

iii) the solution cannot be calculated and does not strongly depend on the data structure. Then we have an ordered structure (e.g. list or vector) containing the enunciates (or pointers to them) and another ordered structure containing the solutions (e.g., list or matrix).

   Example 3. A package for training students in recognizing the style of ancient churches.

The computer tool we have developed is of type iii).

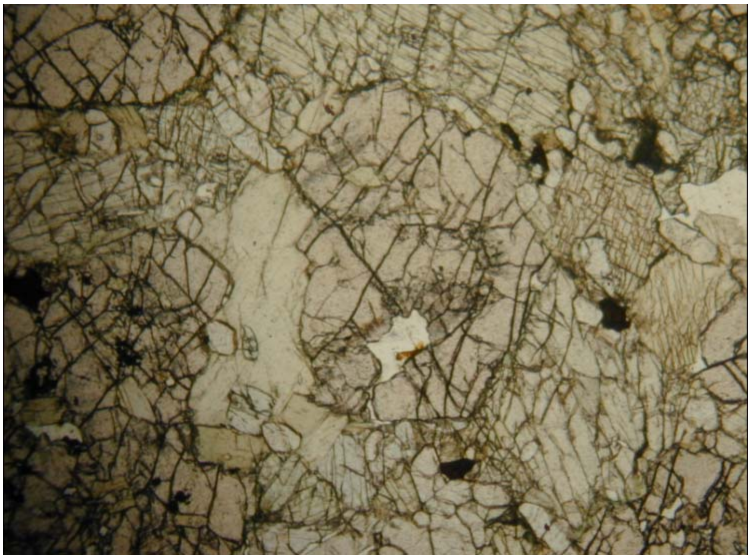


Fig. 1. Photomicrograph of a thin section of granulite (width of field view: 10mm)

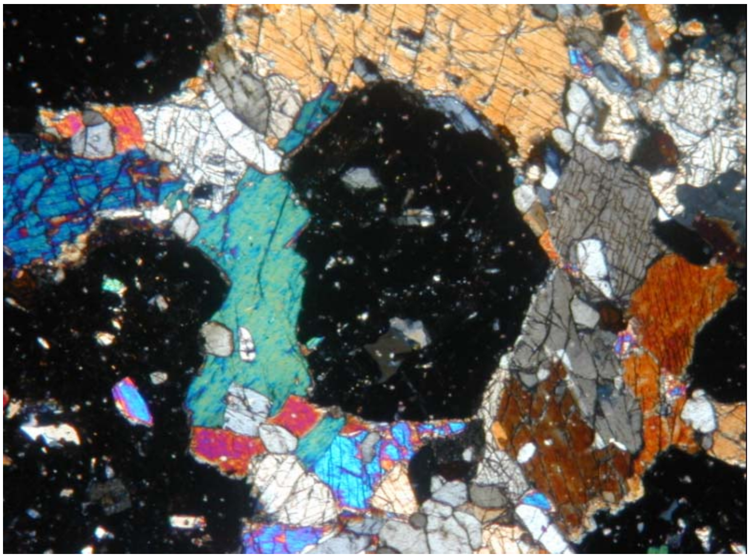


Fig. 2. Photomicrograph of the same thin section of granulite of Figure 1 using crossed nicols (width of field view: 10mm)

1.5 Some Elementary Notions About Petrology

The main characteristic of metamorphic rocks is their texture (texture is understood as the relation of size, shape, and disposition among minerals in a rock) [12,13].

Two kinds of photomicrographies of a thin section of a rock can be taken using a petrographic microscope: “normal” photomicrographies and photomicrographies taken using “crossed nicols” (see figures 1 and 2). The “crossed nicols” is an accessory of the petrographic microscope that allows to analyze the sample under polarized light (which normally enhances the colors and makes the texture clearer).

We have obtained photomicrographies with and without crossed nicols of different samples of the main metamorphic rocks: hornfels, marble, quartzite, granulite, slate, phylite, mylonite, schist, gneiss and migmatite.

We had already faced the problems of handling with *Maple* images stored using other applications when we developed a knowledge-based system for house layout selection [14] (the criteria that lead to determine which house layout scheme fits the requirements of each case best are based on the climate, the building site and the needs of the particular group of occupants).

2 The System’s Design

We have successfully used Gröbner bases to perform knowledge extraction and consistency checking in expert systems based on classic bivalued and modal multi-valued logics [15,16,17]. An overview of our line of research can be found in [18].

But, in this case, all the information has been previously organized and stored by the expert, and the algorithmic nature of the problem makes it unnecessary to perform knowledge extraction and consistency checking in a rule-based standard format. The solution only has to be searched for (and compared with the user’s guess). Anyway, we think this tool can still be denoted *expert system*, as it behaves as an expert in metamorphic rocks classifying.

$$\begin{cases} \begin{array}{c} \textit{foliated}\\ \textit{or}\\ \textit{banded} \end{array} & \begin{cases} \textit{medium grain} & \textit{splintery} \qquad \textbf{hornfels} \\ \textit{coarse grain} & \begin{cases} \textit{main min. calcite} & \textbf{marble} \\ \textit{main min. quartz} & \textbf{quartzite} \\ \textit{main min.pyrox.\&feld.} & \textbf{granulite} \end{cases} \end{cases} \\ \begin{array}{c} \textit{not (foliated}\\ \textit{or banded)} \end{array} & \begin{cases} \textit{very fine grain} & \textbf{slate} \\ \textit{fine grain} & \begin{cases} \textit{green color} & \textbf{phyllite} \\ \textit{pale color} & \textbf{mylonite} \end{cases} \\ \textit{medium grain} & \textbf{schist} \\ \textit{coarse grain} & \begin{cases} \textit{distinct banding} & \textbf{gneiss} \\ \textit{streaky banding} & \textbf{migmatite} \end{cases} \end{cases} \end{cases}$$

Fig. 3. Classiffying common metamorphic rocks

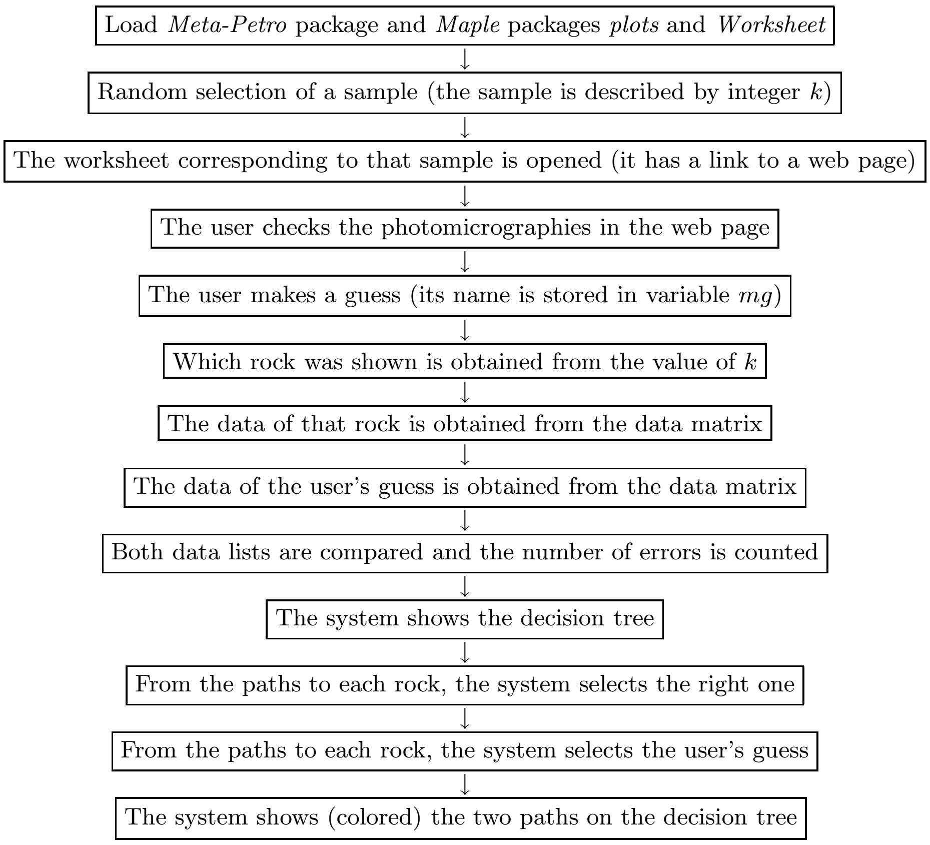

Load *Meta-Petro* package and *Maple* packages *plots* and *Worksheet*
Random selection of a sample (the sample is described by integer k)
The worksheet corresponding to that sample is opened (it has a link to a web page)
The user checks the photomicrographies in the web page
The user makes a guess (its name is stored in variable mg)
Which rock was shown is obtained from the value of k
The data of that rock is obtained from the data matrix
The data of the user's guess is obtained from the data matrix
Both data lists are compared and the number of errors is counted
The system shows the decision tree
From the paths to each rock, the system selects the right one
From the paths to each rock, the system selects the user's guess
The system shows (colored) the two paths on the decision tree

Fig. 4. Algorithm of the whole process

The system is based on the flow chart (tree) in Figure 3, that can be found in [12].

The system should be able not only to give the right solution, but also to show the differences between the user's guess and the right solution. This information can be shown both in matrix format and coloring both paths in a plot of the tree used to classify the rocks.

Decision trees are normally used in Petrology. Nevertheless, in other branches of Geology like Paleontology matrices are used, for instance, to classify fossils according to the presence or absence of certain characteristics [19].

We have decide to use internally matrices and to allow the user to see the data on both formats (as matrices and as decision trees).

The system has the possibility to be easily adapted or enlarged by introducing new or different samples of the types of rocks considered. For instance, a teacher could change the photomicrographies of samples according to his students' needs (if he had the media to obtain photomicrographies). Increasing the number of types of rocks considered would require a little work from the designers.

The system uses *Maple*'s packages `Worksheet` and `plots`. The kind of data checked are stored in list `LD`:

$$[(\mathit{foliated\ or\ banded}),\ \mathit{grain\ size},\ \mathit{splintery},\ \mathit{main\ minerals},\ \mathit{color},\ \mathit{banding}]$$

The names of the rocks are stored in list `LR` and their corresponding data in a matrix of symbols (`DM`) which i-th row contains the data corresponding to the i-th element of `LR` (*DA* means "doesn't apply"):

$$[\mathit{hornfels},\ \mathit{marble},\ \mathit{quartzite},\ \mathit{granulite},\ \mathit{slate},\ \mathit{phyllite},\ \mathit{mylonite},\ \mathit{schist},\ \mathit{gneiss},\ \mathit{migmatite}]$$

$$\begin{bmatrix}
\mathit{not\_fb} & \mathit{medium} & \mathit{splintery} & DA & DA & DA \\
\mathit{not\_fb} & \mathit{coarse} & DA & \mathit{calcite} & DA & DA \\
\mathit{not\_fb} & \mathit{coarse} & DA & \mathit{quartz} & DA & DA \\
\mathit{not\_fb} & \mathit{coarse} & DA & \mathit{pyrox\_and\_feld} & DA & DA \\
\mathit{fb} & \mathit{very\_fine} & DA & DA & DA & DA \\
\mathit{fb} & \mathit{fine} & DA & DA & \mathit{green} & DA \\
\mathit{fb} & \mathit{fine} & DA & DA & \mathit{pale} & DA \\
\mathit{fb} & \mathit{medium} & DA & DA & DA & DA \\
\mathit{fb} & \mathit{coarse} & DA & DA & DA & \mathit{distinct} \\
\mathit{fb} & \mathit{coarse} & DA & DA & DA & \mathit{streaky}
\end{bmatrix}$$

The algorithm followed is shown in Figure 4.

3 Running the System

In the main worksheet, the user can ask *Maple*:

- to show the list of data of a certain rock:
  ```
  > dataRock(mylonite);
  ```

$$\begin{bmatrix}
(\mathit{foliated\ or\ banded}) & \mathit{grain\_size} & \mathit{splintery} & \mathit{main\ minerals} & \mathit{color} & \mathit{banding} \\
\mathit{fb} & \mathit{fine} & DA & DA & \mathit{pale} & DA
\end{bmatrix}$$

- to compare the data of two rocks:
  ```
  > compare(quartzite,gneiss);
  ```

$$\begin{bmatrix}
(\mathit{foliated\ or\ banded}) & \mathit{grain\_size} & \mathit{splintery} & \mathit{main\ minerals} & \mathit{color} & \mathit{banding} \\
\mathit{not\_fb} & \mathit{coarse} & DA & \mathit{quartz} & DA & DA \\
\mathit{fb} & \mathit{coarse} & DA & DA & DA & \mathit{distinct}
\end{bmatrix}$$

Procedure `randRock()` randomly chooses a sample of metamorphic rock and opens the *Maple* worksheet corresponding to that sample. This worksheet contains a link to a web page (see Figure 5) that shows two photomicrographies of the sample (a "normal" one and another one taken using crossed nicols).

After looking at the photomicrographies shown by the system and analyzing them, the user can:

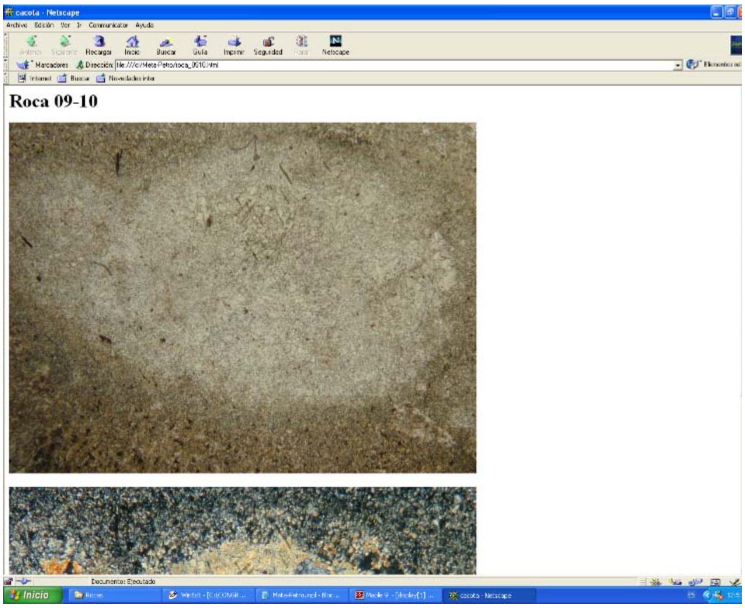


Fig. 5. html page corresponding to one of the samples of hornfels

- introduce a guess to the system:
  `> myGuess(marble):`
- to ask for the data of the user's guess:
  `> myGuessData();`

$$\left[\begin{array}{cccccc} (\textit{foliated or banded}) & \textit{grain\_size} & \textit{splintery} & \textit{main minerals} & \textit{color} & \textit{banding} \\ \textit{not\_fb} & \textit{coarse} & \textit{DA} & \textit{calcite} & \textit{DA} & \textit{DA} \end{array}\right]$$

- to ask for the right solution:
  `> solution();`

$$\textit{hornfels}$$

- to ask for the data of the right solution:
  `> solutionData();`

$$\left[\begin{array}{cccccc} (\textit{foliated or banded}) & \textit{grain\_size} & \textit{splintery} & \textit{main minerals} & \textit{color} & \textit{banding} \\ \textit{not\_fb} & \textit{medium} & \textit{splintery} & \textit{DA} & \textit{DA} & \textit{DA} \end{array}\right]$$

- to compare the data of the user's guess with those of the right solution:
  `> compare(mg,solution());`

$$\left[\begin{array}{cccccc} (\textit{foliated or banded}) & \textit{grain\_size} & \textit{splintery} & \textit{main minerals} & \textit{color} & \textit{banding} \\ \textit{not\_fb} & \textit{coarse} & \textit{DA} & \textit{calcite} & \textit{DA} & \textit{DA} \\ \textit{not\_fb} & \textit{medium} & \textit{splintery} & \textit{DA} & \textit{DA} & \textit{DA} \end{array}\right]$$

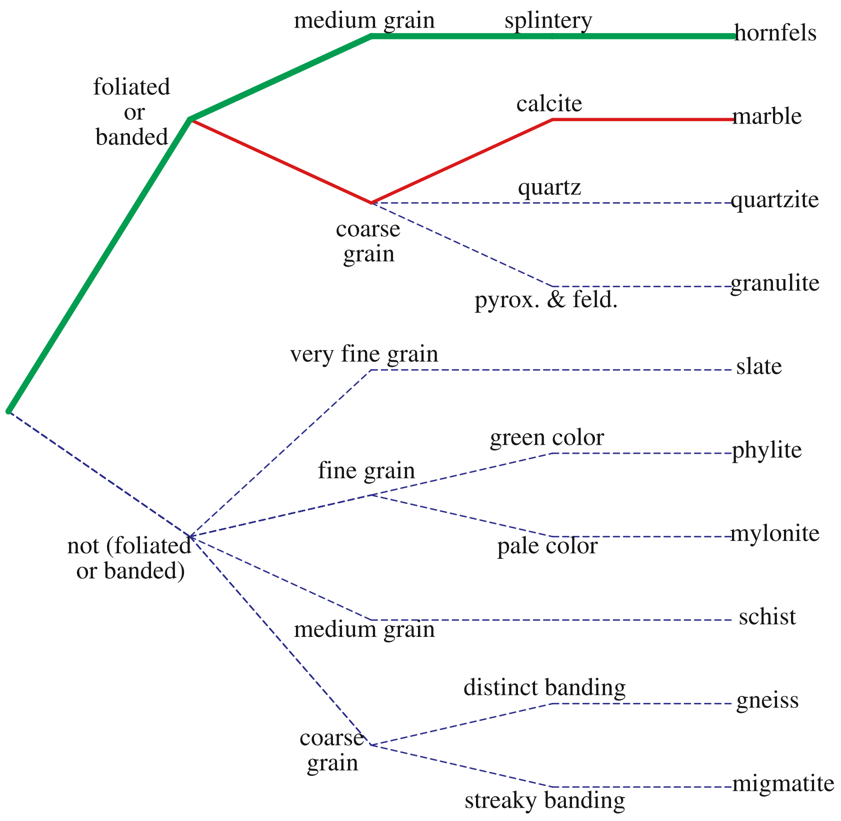


Fig. 6. Automatically generated tree showing the path to the right rock (hornfels) in green and the the path to the user's guess (marble) in red

- to compare the data of a certain rock with those of the right solution:
  `> compare(mylonite,solution());`

$$\left[\begin{array}{cccccc} (\mathit{foliated\ or\ banded}) & \mathit{grain\_size} & \mathit{splintery} & \mathit{main\ minerals} & \mathit{color} & \mathit{banding} \\ \mathit{fb} & \mathit{fine} & \mathit{DA} & \mathit{DA} & \mathit{pale} & \mathit{DA} \\ \mathit{not\_fb} & \mathit{medium} & \mathit{splintery} & \mathit{DA} & \mathit{DA} & \mathit{DA} \end{array}\right]$$

- to count the number of different data between the user's guess and each rock or between the user's guess and the right solution:
  `> differencesWithAll();`

$$\begin{array}{c} \mathit{Differences\ with\ hornfels:\ 3} \\ \mathit{Differences\ with\ marble:\ 0} \\ \mathit{Differences\ with\ quartzite:\ 1} \\ \mathit{Differences\ with\ granulite:\ 1} \\ \mathit{Differences\ with\ slate:\ 3} \\ \mathit{Differences\ with\ phyllite:\ 4} \end{array}$$

```
Differences with mylonite : 4
Differences with schist : 3
Differences with gneiss : 3
Differences with migmatite : 3
```

```
> differencesWithSol();
```

```
Differences with hornfels : 3
```

These data can also be clearly shown on the decision tree. Under the user's request, *Maple* can output:

- the classification tree used:
  `> showTree();`
- the classification tree used, showing the path to the right rock in green and the path to the user's guess in red (hornfels and marble, respectively, in Figure 6):
  `> showDiffInTree();`

4 Conclusions

We think that we have developed an easy to use tool that can be very convenient in Petrology teaching. The main interest of the implementation is the way information is stored and treated.

Future developments would be:

- adding more photomicrographies of new samples to this system
- extending the system to the other types of rocks (plutonic, volcanic, sedimentary)
- developing a GUI based on *Maple*'s *Maplets* (if the users considered it advisable).

The expert system is yet to be experimented in the classroom. We plan to experiment it firstly at the Universidad Complutense de Madrid, that has a *Maple* site license. Remote access to a server running the system in this university is also considered. The package can be freely obtained from the authors.

Acknowledgments

This work was partially supported by the research projects PR3/04-12410 (Universidad Complutense de Madrid, Spain) and MTM2004-03175 (Ministerio de Educación y Ciencia, Spain).

We would also like to thank the "Museo Nacional de Ciencias" for its help in obtaining the thin sections of samples and the corresponding photomicrographies (more than 50) used, that were produced specifically for this work.

References

1. K. Ratajeski, *Atlas of Igneous and Metamorphic Rocks, Minerals & Textures*, `URL://www.geolab.unc.edu/Petunia/IgMetAtlas/mainmenu.html`
2. E. Roanes Lozano, E. Roanes Macías and L.M. Laita, An accelerated-time simulation of departing passengers' flow in airport terminals, *Math. Comp. Simul.* 67 (2004) 163-172.
3. J. Grabmeier, E. Kaltofen and V. Weispfenning, eds., *Computer Algebra Handbook. Foundations. Applications. Systems* (Springer, Berlin, 2003).
4. `URL://www.maplesoft.com/academic/teaching/index.asp`
5. E. Roanes-Lozano, E. López-Vidriero Jr., L.M. Laita, E. López-Vidriero, V. Maojo and E. Roanes-Macías, An Expert System on Detection, Evaluation and Treatment of Hypertension. in B. Buchberger and J.A. Campbell, eds., *Artificial Intelligence and Symbolic Computation AISC 2004* (Springer-Verlag LNAI 3249, Berlin, 2004) 251-264.
6. Anonymous, GUIKit, *Technical software news (Wolfram Research)*, **2** (2004) 2.
7. `URL://www.abcdatos.com/programas/programa/11853.html`
8. `URL://www.worlddidac.org/award2002/history/1990.htm`
9. F. Díez-Rubio, Resolución de ejercicios de Matemáticas con MathEdu. *Forum Tecnológico*, **7** (2004).
(Available at: `URL://www.addlink.es/PDF/AGDWeb654.pdf`)
10. R. Moriyón, Aplicaciones altamente interactivas para el aprendizaje de las Matemáticas (Highly interactive applications for mathematics learning), in M. de Guzmán, E. Roanes-Lozano, P. Ortega and J.P. Garbayo, eds., *International Simposium "Mathematics and New Technologies: What to Learn, How to Teach"* (Universidad Complutense de Madrid, Madrid, 2004).
11. R. Fuster, Exámenes aleatorios con LaTeX, in *Terceras Jornadas de Innovación Docente: La enseñanza de las Matemáticas* (Universidad Politécnica de Valencia, Valencia, 2002).
(Abstract available at:
`URL://www.dma.upv.es/jornadas/jornadasdoc2002/jdoc02/indexJD.htm`)
12. C. Gillen, *Metamorphic geology* (George Allen & Unwin., London, 1982).
13. R. Mason, *Petrology of the metamorphic rocks* (George Allen & Unwin., London, 1978).
14. A. González-Uriel and E. Roanes-Lozano, A knowledge-based system for house layout selection. *Math. Comp. Simul.* **66-1** (2004) 43-54.
15. E. Roanes Lozano, L.M. Laita and E. Roanes-Macías, A Polynomial Model for Multi-Valued Logics with a Touch of Algebraic Geometry and Computer Algebra. *Math. Comp. Simul.* **45-1,2** (1998) 83-99.
16. L.M. Laita, E. Roanes Lozano, V. Maojo, E., Roanes Macías, L. de Ledesma and L. Laita, An Expert System for Managing Medical Appropriateness Criteria based on Computer Algebra Techniques. *Comp. Math. with Appl.* **42-12** (2001) 1505-1522.
17. C. Pérez-Carretero, L.M. Laita, E. Roanes-Lozano, L. Lázaro, J. González-Cajal and L. Laita, A Logic and Computer Algebra-Based Expert System for Diagnosis of Anorexia. *Math. Comp. Simul.* **58-3** (2002) 183-202.
18. E. Roanes Lozano, L.M. Laita, E. Roanes Macías, V. Maojo, S. Corredor, A. de la Vega and A. Zamora, A Groebner Bases-Based Shell for Rule-Based Expert Systems Development, *Exp. Syst. with Appl.* **18/3** (2000) 221-230.
19. A.M. Simonetta and S.C. Morris, eds., *The early evolution of Metazoa and the significance of problematic taxa* (Cambridge University Press, Cambridge, 1989).

The Diamond Operator – Implementation of Exact Real Algebraic Numbers*

Susanne Schmitt

MPI für Informatik, Saarbrücken
`sschmitt@mpi-sb.mpg.de`

Abstract. The LEDA number type `real` is extended by the diamond operator, which allows to compute exactly with real algebraic numbers given as roots of polynomials. The coefficients of these polynomials can be arbitrary real algebraic numbers. The implementation is presented and experiments with two other existing implementations of real algebraic numbers (CORE, EXACUS) are done.

1 Introduction

Real algebraic numbers are real roots of polynomials with integral coefficients. Exact and efficient computation with real algebraic numbers is a crucial part of exact computational geometry [12].

In this paper we describe an implementation of exact real algebraic numbers and compare it with existing real algebraic number types in CORE [2] and EXACUS [5]. The main features of the new implementation are that

– it allows computation (not only comparison) with real algebraic numbers given by polynomials, which is not supported by EXACUS,
– it allows real algebraic numbers given by polynomials which have real algebraic numbers as coefficients, which is not allowed in CORE and in EXACUS.

For a formal definition of real algebraic expressions, we distinguish between the expression E and its value $v(E)$, which is a real algebraic number.

(1) Any integer z is a real algebraic expression. The integer is also the value of the expression: $v(z) = z$.
(2) If E_1 and E_2 are real algebraic expressions, so are $E_1 + E_2$, $E_1 - E_2$, $E_1 \cdot E_2$, E_1/E_2, and $\sqrt[k]{E_1}$, where $k \geq 2$ is an integer. The value $v(\sqrt[k]{E_1})$ is undefined if k is even and $v(E_1)$ is negative. The value $v(E_1/E_2)$ is undefined, if $v(E_2) = 0$. The value of all expressions is undefined, if the value of one of the operands is undefined. Otherwise the value $v(E_1 + E_2)$, $v(E_1 - E_2)$, $v(E_1 \cdot E_2)$, and $v(E_1/E_2)$ is the sum, the difference, the product and the quotient of $v(E_1)$ and $v(E_2)$, respectively and the value $v(\sqrt[k]{E_1})$ is the k-th root of $v(E_1)$.

* Partially supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces).

(3) If $E_d, E_{d-1}, \ldots, E_0$ are real algebraic expressions and j is a positive integer with $0 \leq j \leq d$, then $\diamond(j, E_d, E_{d-1}, \ldots, E_0)$ is a real algebraic expression. If the values $v(E_i)$ are defined, the value $v(\diamond(j, E_d, E_{d-1}, \ldots, E_0))$ is the j-th smallest real root of the polynomial $v(E_d)X^d + v(E_{d-1})X^{d-1} + \ldots + v(E_0)$, if the polynomial has at least j real roots. Otherwise, the value is undefined.

Real algebraic expressions restricted to (1) and (2) are implemented for example in the number type `real` in LEDA, and in CORE. The **diamond operator** $\diamond(\ldots)$ has first been implemented in EXT [6], a LEDA `real` extension number type. EXT is a number package for its own which has been developed for testing the diamond operator. Most parts of EXT are integrated in the LEDA version 5.0.

The EXT implementation is done with the separation bound approach. Here the real algebraic number is approximated with increasing precision as long as the sign is not clear from the interval containing the exact number. This process stops when the length of the interval reaches a specific bound, given by the separation bound. The key issue for that approach is to find good (i.e. large and easily computable) separation bounds. There are several separation bounds, some of them are not in general comparable [1], [11], [15], [19].

The separation bound approach is also implemented in CORE. The difference to the EXT implementation is that CORE does not allow real algebraic expressions as coefficients of the diamond operator (called RootOf in CORE).

EXACUS represents general real algebraic numbers by a defining polynomial and an isolating interval which contains only this root of the polynomial. Comparison is easy if the intervals do not intersect. If they intersect, one has to do further refinement and possibly one has to compute the gcd of the polynomials to decide whether they have a common factor. To do this efficiently, the polynomials are restricted to polynomials with integral coefficients. Comparison of two algebraic numbers is fast in general. Computing with these algebraic numbers gets difficult, as one has to provide always an integral defining polynomial. Therefore these representations often do not allow computation with real algebraic numbers.

The real algebraic numbers of Guibas, Karavelas and Russel [7] or of Rioboo et al. [13], [17] are implemented with a similar concept than the EXACUS real algebraic numbers.

Emiris and Tsigaridas [4] present an algorithm for exact comparison of the real roots of two polynomials of degree ≤ 4. Their method relies on isolating intervals for the representation of the roots in a specific way and on precomputing Sturm sequences to minimize computational effort.

2 Implementation

An EXT `real` is represented by the real algebraic expression which defines it and an open interval I that contains its exact value. The interval is given by a `bigfloat` (floating point with arbitrary precision) approximation and an error. When the sign of a `real` x needs to be determined, the data type uses the separation bound approach and first computes a number q, such that $|x| \leq 2^{-q}$ implies $x = 0$. In the current EXT version, this bound is either the improvement of the BFMSS bound [1], [15], [19] or the degree measure bound from [11], if this is larger.

Using floating point arithmetic with arbitrary-length mantissa, the data type then computes an interval I of maximal length 2^{-q} that contains x. If I contains zero, then x itself is equal to zero. Otherwise, the sign of any point in I is returned as the sign of x.

Two shortcuts are used to speed up the computation of the sign. Firstly, if the initial interval approximation already suffices to determine the sign, no further `bigfloat` approximation is computed at all. Secondly, the `bigfloat` approximation is first computed only with small precision. The precision is then roughly doubled until either the sign can be decided (i.e., if the current approximation interval does not contain zero) or the full precision 2^{-q} is reached. This procedure makes the sign computation of a `real` x *adaptive* in the sense that the running time of the sign computation depends on the complexity of x.

The diamond operator is integrated into the EXT `real` number package. The two important steps of the implementation of the diamond operator are

– to compute an isolating interval which contains only the intended real root of the given polynomial
– to approximate the real root with increasing precision.

The isolating interval is computed with the method based on Descartes rule of sign[1] as described for example in [8], [18], [9] (see also [14]). Applied on the polynomial with exact `real` coefficients, this algorithm performs sign computation of possibly large `real` expressions. To speed up the isolating interval computation, the EXT package replaces the exact coefficients of the polynomial by their interval approximation and applies the Descartes real root isolation method for polynomials given by interval coefficients [8], [18]. This results in a much faster algorithm, which succeeds in many cases to compute isolating intervals. However there are cases where the interval root isolation method cannot compute isolating intervals, because the sign of an interval cannot be determined. In these cases the `real`s use the exact root isolation method. In [3] the authors give a complete and exact Descartes method which does not need to rely on exact arithmetic as in the implementation above.

2.1 Approximation of the Root

The EXT `real` number type uses the LEDA `bigfloat` package to approximate a real algebraic number with a given error bound. The approximation of a number given by the $\diamond$-expression is done in two steps. First a `bigfloat` polynomial is created which has a root near the exact root of the exact polynomial. Then the Newton method is applied on that `bigfloat` polynomial. In this subsection we consider the first step. Let

[1] The disadvantage of this method is that it allows only isolated interval computation for square free polynomials, but there are methods to circumvent this problem.

$$P(x) = \sum_{i=0}^{d} a_i x^i$$

be the polynomial with the exact `real` coefficients a_i. Consider a `bigfloat` approximation

$$\tilde{P}(x) = \sum_{i=0}^{d} \widetilde{a_i} x^i$$

of that polynomial, that is, the coefficients $\widetilde{a_i}$ of $\tilde{P}$ are `bigfloat` approximations of the coefficients a_i of P:

$$|\widetilde{a_i} - a_i| \leq 2^{-q}.$$

The polynomial $\tilde{P}$ can also be written as a sum of two polynomials $\tilde{P}(x) = P(x) + E(x)$, where the second polynomial contains the errors:

$$E(x) = \sum_{i=0}^{d} e_i x^i \quad \text{with } |e_i| \leq 2^{-q}.$$

Let α be a root of the polynomial $P(x)$. The goal is to compute an approximation $\overline{\alpha}$ of this root with an error $|\overline{\alpha} - \alpha| \leq 2^{-p}$.

Suppose we know in advance that α lies in the interval (a, b) and that $P(x)$ has no double roots. Using bisection we can shrink the isolating interval in such a way that the first derivative of P has no root in (a, b). Let $m := \max\{|a|, |b|\}$.

Choosing q large enough, we can show that the polynomial $\tilde{P}(x)$ has a simple root near α:

Lemma 1. *If q satisfies the conditions*

a) $2^{-q} \sum_{i=0}^{d} m^i < \min\{|P(a)|, |P(b)|\}$,
b) $2^{-q} \sum_{i=0}^{d-1} (i+1) m^i < \min_{c \in (a,b)}\{|P'(c)|\}$,

then $\tilde{P}(x)$ has exactly one simple root $\widetilde{\alpha}$ in the interval (a, b).

Proof. Let $\xi \in (a, b)$. Then $|P'(\xi)| \geq \min_{c \in (a,b)}\{|P'(c)|\}$. Condition b) then gives

$$|P'(\xi)| > 2^{-q} \sum_{i=0}^{d-1} (i+1) m^i.$$

The right hand side is an estimate for $|E'(\xi)|$:

$$|E'(\xi)| = \left| \sum_{i=0}^{d-1} (i+1) e_{i+1} \xi^i \right| \leq \sum_{i=0}^{d-1} (i+1) |e_{i+1}| |\xi|^i \leq 2^{-q} \sum_{i=0}^{d-1} (i+1) m^i.$$

Together we see that $|P'(\xi)| > |E'(\xi)|$. With $\tilde{P}'(\xi) = P'(\xi) + E'(\xi)$ this shows that $\tilde{P}'(\xi)$ and $P'(\xi)$ have the same sign. Hence, as $P'(x)$ has no root in (a, b), the polynomial $\tilde{P}'(x)$ also has no root in (a, b).

Using Condition a) we see in a similar way as above, that $|E(a)| < |P(a)|$ and $|E(b)| < |P(b)|$, so that the sign of $\tilde{P}(a)$ and $P(a)$ resp. $\tilde{P}(b)$ and $P(b)$ is the same. It follows that $\tilde{P}(x)$ has exactly one simple root in (a, b). □

With the next condition we quantify how close the root $\tilde{\alpha}$ is to the root α of the exact polynomial.

Proposition 1. *Let $p > 0$. If q satisfies the conditions a) and b) from the lemma and*

c) $q \geq p + 1 + \log_2 \left(\sum_{i=0}^{d} m^i \right)$,

then for the root $\widetilde{\alpha}$ of $\tilde{P}(x)$ in (a,b) we have $|\widetilde{\alpha} - \alpha| \leq 2^{-p-1}/M$, where $0 < M < \min_{c \in (a,b)} \{|\tilde{P}'(c)|\}$.

Proof. Using the mean value theorem we get the equation

$$\frac{\tilde{P}(\widetilde{\alpha}) - \tilde{P}(\alpha)}{\widetilde{\alpha} - \alpha} = \tilde{P}'(\xi),$$

where ξ is a number between $\widetilde{\alpha}$ and α. From this equation we get, using $\tilde{P}(\widetilde{\alpha}) = 0$ and $P(\alpha) = 0$,

$$|\widetilde{\alpha} - \alpha| = \frac{|\tilde{P}(\alpha)|}{|\tilde{P}'(\xi)|} = \frac{|E(\alpha)|}{|\tilde{P}'(\xi)|}.$$

Now we estimate

$$|E(\alpha)| \leq \sum_{i=0}^{d} |e_i||\alpha|^i \leq 2^{-q} \sum_{i=0}^{d} |\alpha|^i \leq 2^{-q} \sum_{i=0}^{d} m^i.$$

If we choose q as in Condition c), we get

$$|\widetilde{\alpha} - \alpha| \leq 2^{-p-1} \frac{1}{|\tilde{P}'(\xi)|}.$$

As $\tilde{P}'(x)$ has no roots in (a,b), we can estimate the minimum M of $\tilde{P}'(x)$ in (a,b) and get $|\widetilde{\alpha} - \alpha| \leq 2^{-p-1}/M$. □

If we now choose

$$r \geq -\log_2(2^{-p} - 2^{-p-1}/M)$$

and apply the Newton method to compute an approximation $\overline{\alpha}$ of the root $\widetilde{\alpha}$ of $\tilde{P}(x)$ up to an error $|\overline{\alpha} - \widetilde{\alpha}| \leq 2^{-r}$, we get the desired approximation:

$$|\overline{\alpha} - \alpha| \leq |\overline{\alpha} - \widetilde{\alpha}| + |\widetilde{\alpha} - \alpha| \leq 2^{-r} + 2^{-p-1}/M \leq 2^{-p}.$$

Summarizing this subsection, to get the desired approximation of the exact root, we need to find approximations of the coefficients of the exact polynomial such that the errors satisfy conditions a), b) and c), and we need a lower bound for the minimal absolute value of the derivative of the approximating polynomial in the interval.

2.2 Newton Method for `bigfloat` Polynomials

The implementation of the Newton method for `bigfloat` polynomials deserves some explanation. Let $\widetilde{\alpha}$ be the exact root of the `bigfloat` polynomial $\tilde{P}(x)$ in the interval (a, b). We consider the Newton iteration function

$$F(x) = x - \frac{\tilde{P}(x)}{\tilde{P}'(x)}$$

First of all we have to ensure that the Newton method converges.

Lemma 2. *Let $b - a \leq 2^{-t}$.*

(1) If

$$\frac{\max_{x\in(a,b)}\{|\tilde{P}''(x)|\}\max_{x\in(a,b)}\{|\tilde{P}(x)|\}}{(\min_{x\in(a,b)}\{|\tilde{P}'(x)|\})^2} =: c < 1$$

then the Newton method with any start value $x \in (a, b)$ converges linearly and

$$|F(x) - \widetilde{\alpha}| \leq 2^{-t+\log c}.$$

(2) If

$$\frac{\max_{x\in(a,b)}\{|\tilde{P}'''(x)|\}\max_{x\in(a,b)}\{|\tilde{P}(x)\}}{(\min_{x\in(a,b)}\{|\tilde{P}'(x)|\})^2} + \frac{\max_{x\in(a,b)}\{|\tilde{P}''(x)|\}}{\min_{x\in(a,b)}\{|\tilde{P}'(x)|\}}$$

$$-2\frac{(\max_{x\in(a,b)}\{|\tilde{P}''(x)|\})^2\max_{x\in(a,b)}\{|\tilde{P}(x)|\}}{(\min_{x\in(a,b)}\{|\tilde{P}'(x)|\})^3} =: C < 2^t$$

then the Newton method with any start value $x \in (a, b)$ converges quadratically and

$$|F(x) - \widetilde{\alpha}| \leq 2^{-2t-1+\log C}.$$

Proof. These conditions come from the conditions for the convergence rates of the Newton method. The proofs can be found in any numerical mathematics book. □

At the start of the Newton method, we use bisection to shrink the isolating interval until condition (1) or (2) is satisfied. Then we can assure that the Newton method converges.

Now we look at the particular Newton steps. At the beginning of a Newton step we have an approximation x_i of $\widetilde{\alpha}$ with $|x_i - \widetilde{\alpha}| \leq 2^{-l}$. The next approximation is computed in the Newton step $x_{i+1} = F(x_i)$. If everything is computed exactly, then $|x_{i+1} - \widetilde{\alpha}| \leq 2^{-(l'+1)}$.

Here $l' + 1 > l$. The choice of $l' + 1$ depends whether condition (1) or (2) is satisfied: If condition (1) is satisfied, then $l' + 1 = l - \log c$. If condition (2) is satisfied, then $l' + 1 = 2l + 1 - \log C$. (The $+1$ is chosen to deal with the inexact computation with `bigfloat` numbers.)

The values $\tilde{P}(x_i)$ and $\tilde{P}'(x_i)$ are computed exactly. This is possible, because there is no division. If the polynomial evaluation would be done inexactly, one would need lower bounds on the absolute value of the results for the error bounds. Such bounds can not be computed for $\tilde{P}(x_i)$, as this value should is very small.

The `bigfloat` division $\frac{\tilde{P}(x_i)}{\tilde{P}'(x_i)}$ is performed with absolute error less than $2^{-(l'+3+\log_2(|\tilde{P}(x_i)|)-\log_2(|\tilde{P}'(x_i)|))}$.

The subtraction is also done exactly. This could be done inexactly providing an error bound, but there is not much profit and it makes the analysis easier if there is no computation error.

The result is a `bigfloat` approximation $\widetilde{x_{i+1}}$ of x_{i+1} with absolute error $|\widetilde{x_{i+1}}-x_{i+1}| \leq 2^{-(l'+2)}$. This value is rounded to absolute precision $2^{-(l'+2)}$, such that the result has an absolute error $|\text{round}(\widetilde{x_{i+1}}) - x_{i+1}| \leq 2^{-(l'+1)}$. Together with $|x_{i+1} - \widetilde{\alpha}| \leq 2^{-(l'+1)}$ this gives $|\text{round}(\widetilde{x_{i+1}}) - \widetilde{\alpha}| \leq 2^{-l'}$.

3 Experiments

This section contains experiments with real algebraic numbers given by polynomial in EXT, EXACUS and CORE. More experiments can be found on `http://www.mpi-sb.mpg.de/projects/EXACUS/leda_extension/experiments.html`.

The two systems EXACUS and CORE have been chosen for the first set of experiments, because their understanding of arithmetic with exact real algebraic numbers is similar. There are other approaches to exact real algebraic computation. SYNAPS [16] for example assumes that the user provides a bound on the minimal distance between two roots of a polynomial. Of course such a bound can be computed in a similar way as the separation bound, but this system is not designed with the intention to compute this bound internally.

We did no comparison with systems providing real root isolation methods. Computing an isolating interval is only one (nevertheless important) part of the EXT reals. In this set of experiments we wanted to test the overall performance of computing with real algebraic numbers.

The experiments were done on a 850 MHz Pentium III machine. We measured the time needed to generate the real algebraic numbers and to compare them.

CORE uses Sturm sequences to isolate real roots and therefore can handle polynomials with multiple roots. The EXT package and EXACUS both use the Descartes isolating method, which needs square free polynomials. Hence for these packages, a square free test has to be made before the real root isolation. If a polynomial $P(x)$ is not square free, one has to divide $P(x)$ by the gcd of $P(x)$ and $P'(x)$. In the examples however, all randomly generated polynomials were square free. The time of this square free test is also measured.

3.1 Comparison

For the first test cases we generated random polynomials of degree d (for odd d) where the coefficients have bit-size L. The probability that two such polynomials have a common root is negligible. As the degree of the polynomials is odd, they all have at least one real root. The first test checks if the smallest real roots of two such polynomials are equal. As they are not equal, this test should be very fast. The results for degree 5 polynomials with coefficients of growing bit-size are shown in Figure 1.

To test two equal diamond expressions, we again generated random polynomials of odd degree d and coefficient bit-size L. Let $P(x)$ be such a polynomial. Then $Q(x) = P(x) \cdot (x^2 + 1)$ has the same real roots as $P(x)$. The second test cases check if the first real roots of both polynomials are equal. The results for degree 5 polynomials with coefficients of growing bit-size are shown in Figure 2.

In both cases, EXACUS provides the fastest method. For different numbers, the EXT package is also quite efficient. For equal numbers, both methods relying on the separation bound approach need more time than EXACUS.

We did the same experiments for polynomials with different degrees. The results in Figure 3 and Figure 4 show that both separation bound methods have problems with polynomials of larger degree. The bad running time of the EXT

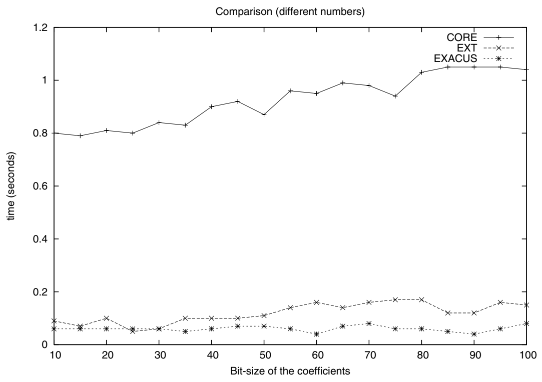


Fig. 1. 100 comparisons of (different) real roots of Polynomials of degree 5

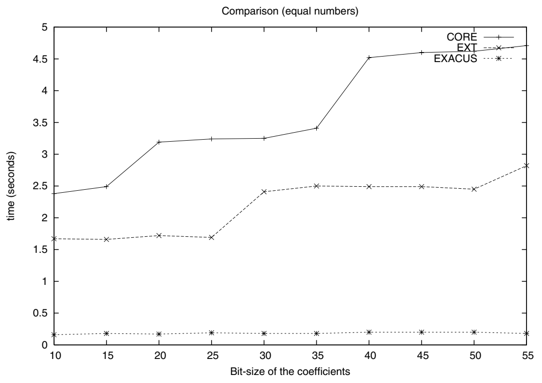


Fig. 2. 200 comparisons of (equal) real roots of Polynomials of degree 5

real numbers in Figure 3 comes from the failure of the interval Descartes method. Here the isolation method has to fall back on exact computation with real algebraic numbers for isolating the real root. This will be improved in the future, using [3]. However, current applications of the diamond operator in EXACUS only need polynomials of degree ≤ 10.

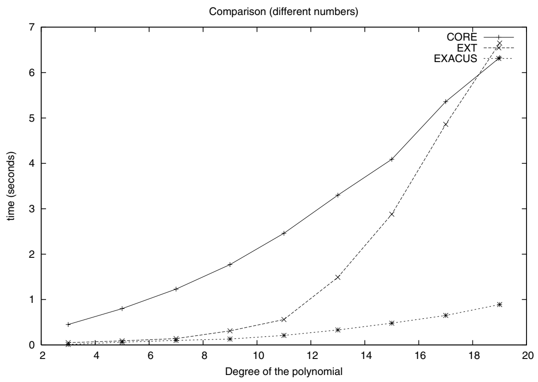


Fig. 3. 100 comparisons of (different) real roots of Polynomials with coefficients of bit-size 10

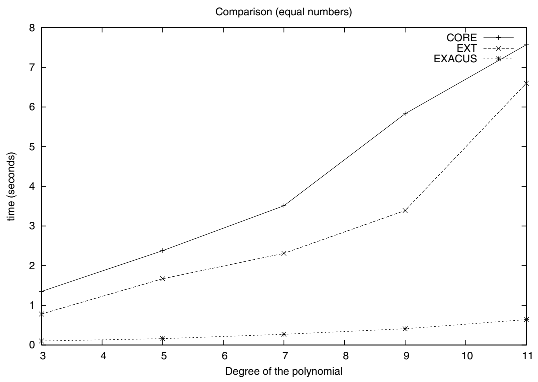


Fig. 4. 200 comparisons of (equal) real roots of Polynomials with coefficients of bit-size 10

3.2 Computation

In CORE and EXT one can perform computations with real roots of polynomials. These computations include $+, -, \cdot, /$ and $\sqrt[k]{\ }$. In EXACUS the real algebraic numbers resulting from roots of polynomials do not support computation. However, it is possible to do computation in a restricted sense if one can find an integral defining polynomial.

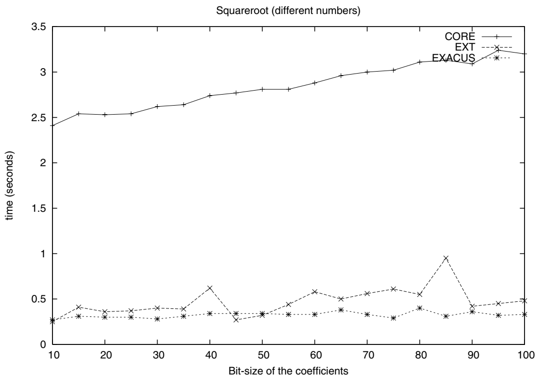


Fig. 5. 100 comparisons of (different) squareroots of real roots of polynomials of degree 5

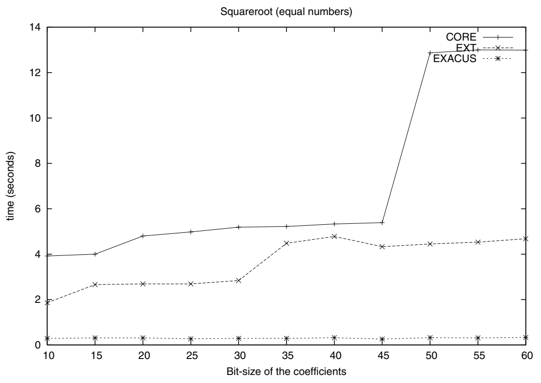


Fig. 6. 200 comparisons of (equal) squareroots of real roots of polynomials of degree 5

For the tests we considered a polynomial $P(x)$ with a positive real root α and a polynomial $Q(x)$ with a positive real root β. The test was to check if $\sqrt{\alpha} = \beta$.

To find an integral defining polynomial for $\sqrt{\alpha}$ consider the polynomial $R(x) = P(x^2)$. The real roots of $R(x)$ are the square roots of the positive real roots of $P(x)$. Using this polynomial it is possible to perform the tests in EXACUS. The time for generating $R(x)$ and its roots was also measured in the EXACUS tests.

Figure 5 and Figure 6 contain the test for polynomials of degree 5 with coefficients of growing bit-size. Again for different numbers the EXT package is as efficient as the EXACUS package. When the real algebraic numbers are equal, the two separation bound methods need more time.

3.3 Real Algebraic Coefficients

Unlike CORE and EXACUS, EXT can handle roots of polynomials as `reals`. That means that one can do exact computation with those numbers and can take those numbers also as coefficients of polynomials of other $\diamond$-operators.

In the example we generated the first real root α of the polynomial x^3+2x-1. Then we generated polynomials of odd degree, as before, and added α to the coefficient of x. The tests were performed on degree 5 polynomials with coefficient of growing bit-size. The computation time (in seconds) of the equality test for different numbers and for equal numbers are given in Figure 7.

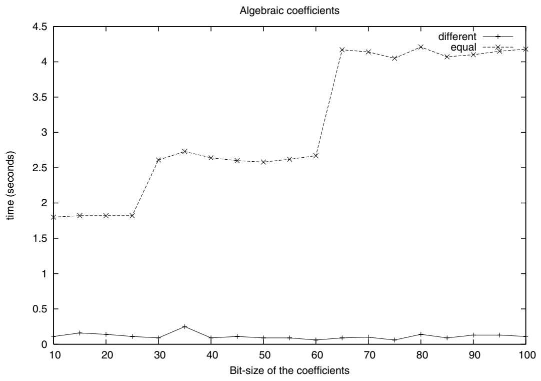


Fig. 7. 100 comparison of real roots of polynomials of degree 5 with algebraic coefficients

3.4 Conclusion

The experiments indicate that the separation bound approach for the diamond operator implemented in EXT is efficient for comparisons of different numbers of degree ≤ 10. In the degenerate cases (that is, when the numbers are equal), the separation bound approaches both need more time for comparisons.

It is interesting to note that also in the difficult case (Figure 5 and Figure 6), where the integral defining polynomial has to be computed, the isolating interval method from EXACUS has the best running time. This indicates that the real root isolation method should be preferred to the separation bound method, when integral defining polynomials can be found.

Future experiments, also with other real algebraic number packages, should give more insight in the behavior of the different methods. Especially the behavior of iterated diamond expressions should be considered in more detail.

References

1. C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A Separation Bound for Real Algebraic Expressions. In *Algorithms - ESA 2001 (LNCS 2161)*, pages 254-265, 2001.
2. CORE (A core library for robust numeric and geometric computation). `http://www.cs.nyu.edu/exact/core/`
3. A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, and N. Wolpert. An Exact Descartes Algorithm with Approximate Coefficients (extended abstract). To appear in *Proc. 8th Internat. Workshop on Computer Algebra in Scient. Comput. (CASC 2005)*.

4. I. Z. Emiris and E. P. Tsigaridas. Comparing Real Algebraic Numbers of Small Degree. ESA 2004, 652-663. Technical Report, ECG-TR-242200-01, 2003
5. EXACUS (Efficient and Exact Algorithms for Curves and Surfaces). `http://www.mpi-sb.mpg.de/projects/EXACUS/`
6. EXT (LEDA `real` Extended). `http://www.mpi-sb.mpg.de/projects/EXACUS/leda_extension/`
7. L. Guibas, M. I. Karavelas, and D. Russel. A computational framework for handling motion. In *Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments* 2004, 129-141.
8. J. R. Johnson and W. Krandick. Polynomial Real Root Isolation using Approximate Arithmetic. In *International Symposium on Symbolic and Algebraic Computation*, 225-232, 1997.
9. W. Krandick and K. Mehlhorn. New Bounds for the Descartes Method. Technical Report, Drexel University DU-CS-04-04, 2004.
10. LEDA (Library of Efficient Data Types and Algorithms). `http://www.mpi-sb.mpg.de/LEDA/`
11. C. Li and C. Yap. A new constructive root bound for algebraic expressions. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 496-505, 2001.
12. C. Li and C. Yap. Recent progress in exact geometric computation. In *Proc. DIMACS Workshop on Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science*, 2001.
13. Z. Ligatsikas and R. Rioboo and M.-F. Roy, Generic computation of the real closure of an ordered field. Mathematics and Computers in Simulation, v.42 n.4-6, p.541-549, 1996
14. B. Mourrain, M. Vrahatis, J. C. Yakoubsohn. On the Complexity of Isolating Real Roots and Computing with Certainty the Topological Degree. J. of Complexity 18(2), 612-640, 2002.
15. S. Pion and C. Yap. Constructive root bound method for k-ary rational input numbers. In *19th ACM Symp. on Comp.Geometry*, 256–263, San Diego, California., 2003.
16. G. Dos Reis, B. Mourrain, R. Rouillier, P. Trebuchet. An environment for symbolic and numeric computation. In Proc. of the International Conference on Mathematical Software 2002, World Scientific, 239-249, 2002.
17. R. Rioboo. Towards faster real algebraic numbers. In *Journal of Symbolic Computation, Special issue: ISSAC 2002* , Volume 36, pages 513 - 533, 2003.
18. F. Rouillier and P. Zimmermann. Efficient Isolation of a Polynomial Real Roots. Journal of Computational and Applied Mathematics, 162(1): 33-50, 2003
19. S. Schmitt. Improved separation bounds for the diamond operator. Effective Computational Geometry for Curves and Surfaces ECG-TR-363108-01 Report, 2004.
20. C. K. Yap. *Fundamental Problems in Algorithmic Algebra.* Oxford University Press, 1999.

Constructing the Numerical Method for Navier — Stokes Equations Using Computer Algebra System

Leonid Semin and Vasily Shapeev

Institute of Theoretical and Applied Mechanics SB RAS, Novosibirsk, 630090, Russia
l_semin@ngs.ru

Abstract. The present study demonstrates a very helpful role of computer algebra systems (CAS) for deriving and testing new numerical methods. We use CAS to construct and test a new numerical method for solving boundary - value problems for the 2D Navier — Stokes equations governing steady incompressible viscous flows. We firstly describe the core of the method and the algorithm of its construction, then we describe the implementation in CAS for deriving formulas of the method and for testing them, and finally we give some numerical results and concluding remarks.

1 Introduction

Solving the boundary-value problems for Navier — Stokes equations is a complicated task of computational fluid dynamics. Many numerical methods, which demonstrate a good performance when solving problems for other equations, do not work when applied to Navier — Stokes equations. That is why there is a permanent search for new methods to solve this problem.

We have proposed the method of collocation and least squares (CLS) for solving this problem. Firstly it was implemented to solve the boundary - value problems for Stokes equations [1]. In the present study we shall briefly describe the application of the CLS method to Navier — Stokes equations. The method has a number of advantages: the continuity equation is satisfied exactly, the boundary conditions are easily approximated, the method can easily be generalized to the case of three dimensions and to the case of domain with curvilinear boundary without loss of accuracy, the approximation order can be increased just by adding new basis functions. But huge analytical work is needed when constructing the formulas of the CLS method for Navier — Stokes equations. To great extent such work can be carried out by computer algebra systems (CAS) [2,3,4,5]. To construct the formulas one needs to perform analytically the following steps: summation of huge expressions in functionals, their differentiation, solving linear algebraic systems in order to obtain coefficients of approximate solution in polynomial basis. CAS can successfully cope with this work. Computer derivation of the formulas of the method gives additional advantages. Optimization with respect to the rate of computation by obtained formulas, analytical and numerical verification of these formulas by using the same or another CAS, conversion of constructed formulas to arithmetic operators of FORTRAN and C programming languages can be made. All these factors facilitate greatly the process and reduces the time for reaching final result. Different modifications of the method are easily available using computer programs in CAS. Modifications related to replacement of basis, collocation nodes, matching conditions etc., can be made just by changing functions involved in the program.

2 Description of the CLS Method and Its Construction Algorithm

The basic idea of the method is to use the collocation method in combination with the least-squares method to obtain numerical solution. We call such a combined method "collocation and least-squares" method (CLS). One of the reasons for using this combination was that the application of the least-squares method often improves the properties of a numerical method. In turn, the collocation method is simple in implementation and gives good results when solving boundary-value problems for ODEs, both linear and nonlinear [6].

Let us consider the boundary-value problem for Navier — Stokes equations:

$$\begin{cases} \mathrm{Re}^{-1} \triangle v_j - v_1 \dfrac{\partial v_j}{\partial x_1} - v_2 \dfrac{\partial v_j}{\partial x_2} - \dfrac{\partial p}{\partial x_j} = f_j, \qquad (x_1, x_2) \in \Omega, \quad j = 1, 2, \\ \operatorname{div} \boldsymbol{v} = 0, \\ \boldsymbol{v}|_{\partial\Omega} = \boldsymbol{V}. \end{cases} \tag{1}$$

Here Re is the Reynolds number, $\partial\Omega$ is the boundary of domain Ω. The approximate solution is found as a piecewise polynomial function on a regular grid. Let us introduce the following denotations: h is the half-width of the cell, (x_{1m}, x_{2m}) are the coordinates of mth cell center, $y_1 = (x_1 - x_{1m})/h$, $y_2 = (x_2 - x_{2m})/h$ are the local coordinates in the cell, $\boldsymbol{u}(y_1, y_2) = \boldsymbol{v}(x_1, x_2)$, $q(y_1, y_2) = hp(x_1, x_2)$. Then equations (1) in local variables can be written as follows:

$$\triangle u_j - \mathrm{Re}\left(h\left(u_1 \frac{\partial u_j}{\partial y_1} + u_2 \frac{\partial u_j}{\partial y_2}\right) + \frac{\partial q}{\partial y_j}\right) = \mathrm{Re}\, h^2 f_j, \qquad j = 1, 2, \tag{2}$$

$$\operatorname{div} \boldsymbol{u} = 0, \tag{3}$$

$$\boldsymbol{u}|_{\partial\Omega} = \boldsymbol{U}. \tag{4}$$

The essential feature of the Navier — Stokes equations consists in the presence of nonlinear terms. Therefore, linearization of these equations is needed to obtain a system of *linear* equations used to find the solution. This is carried out as follows. Let some approximate solution $(\breve{u}_1, \breve{u}_2, \breve{q})$ be known. Let us represent the desired improved solution in the form: $u_1 = \breve{u}_1 + \widehat{u}_1$, $u_2 = \breve{u}_2 + \widehat{u}_2$, $q = \breve{q} + \widehat{q}$. Substituting this representation in equations (2) and neglecting the second-order terms $\widehat{u}_1 \widehat{u}_{j,y1}$, $\widehat{u}_2 \widehat{u}_{j,y2}$, $j = 1, 2$, we obtain the linearized equations:

$$\triangle \widehat{u}_j - \mathrm{Re}\, h(\breve{u}_1 \widehat{u}_{j,y1} + \breve{u}_2 \widehat{u}_{j,y2} + \widehat{u}_1 \breve{u}_{j,y1} + \widehat{u}_2 \breve{u}_{j,y2}) - \mathrm{Re}\, \widehat{q}_{y_j} = F_j, \quad j = 1, 2, \tag{5}$$

where $F_j = \mathrm{Re}(h^2 f_j + h(\breve{u}_1 \breve{u}_{j,y1} + \breve{u}_2 \breve{u}_{j,y2}) + \breve{q}_{y_j}) - \triangle \breve{u}_j$, $v_{k,y} = \frac{\partial v_k}{\partial y}$.

Improvement of solution in each cell is sought for in the form:

$$\begin{pmatrix} \widehat{u}_1 \\ \widehat{u}_2 \\ \widehat{q} \end{pmatrix} = \sum_j a_{jm} \varphi_j, \tag{6}$$

where φ_j are basis vector - functions with three components, m is the cell number.

Coefficients a_{jm} will be determined from the collocation equations and matching conditions on the boundary of adjacent cells or boundary conditions at $\partial\Omega$.

The selected basis functions assume that the components of the velocity vector and the pressure are polynomials. We have implemented a variants of the method where the velocity components are second-order polynomials, and the pressure is either linear function or second order polynomial too. Moreover, it is required that solution (6) satisfies the continuity equation div $\boldsymbol{u} = 0$ due to the selection of basis functions. It means that the continuity equation will be satisfied in the whole domain on any numerical solution found.

For the case of pressure approximation by the second-order polynomial, there are 15 basis functions. Below, we will write down the summation indices in all formulas corresponding to this case.

It is necessary to specify the matching conditions at the intercell boundaries to obtain a unique piecewise polynomial solution. The continuity conditions of the following expressions are considered as matching conditions:

$$\frac{\partial U_n}{\partial n} - \eta p + \eta U_n, \qquad \frac{\partial U_t}{\partial n} + \eta U_t. \tag{7}$$

Here U_n and U_t are the velocity components normal and tangential to the boundary, respectively, n is the outside unit vector normal to the cell, η is a positive parameter. The latter can affect the conditionality of the obtained system of linear algebraic equations and the convergence rate. Generally speaking, these conditions cannot be satisfied along the whole intercell boundary. We satisfy them in terms of least squares at eight points at the cell boundary.

If the cell boundary coincides with the domain boundary, then velocity vector is specified at two points at this boundary. Moreover, the pressure is specified at the lower left corner of the domain.

Collocation equations of (5) at four points inside the cell are added to the matching conditions (boundary conditions). In local coordinates, the matching conditions are specified at points $(\pm 1, \pm\zeta)$, $(\pm\zeta, \pm 1)$, the boundary conditions are specified at points $(\pm 1, \pm\xi)$, $(\pm\xi, \pm 1)$, the collocation equations are specified at points $(\pm\omega, \pm\omega)$. Values ζ, ξ, ω are positive and less than unity. They can be chosen in different manner to obtain a well - conditioned matrix of the system of linear algebraic equations.

As a result, the following system of linear algebraic equations is obtained:

$$\sum_{k=1}^{15} B_{lk} a_{km} = F_l, \qquad l = 1, \ldots, 24. \tag{8}$$

Here the first four equations were obtained from matching conditions (7) or the boundary conditions at the bottom boundary, the next four equations were obtained from the corresponding conditions at the right boundary, equations at $l = 9, \ldots, 12$ were obtained from conditions at the top boundary, and equations at $l = 13, \ldots, 16$ were obtained from the conditions at the left boundary. Equations at other l were obtained from the collocation equations.

The system (8) is overdetermined: it involves more equations than unknowns. Let us consider the following two functionals to define what is considered to be a solution of this system:

$$\varPhi_1 = \sum_{l=1}^{16} (\sum_j B_{lj} a_{jm} - F_l)^2, \qquad \varPhi_2 = \sum_{l=17}^{24} (\sum_j B_{lj} a_{jm} - F_l)^2. \tag{9}$$

The first functional corresponds to the sum of residual squares of equations obtained from the matching or boundary conditions, the second one corresponds to the sum of the residual squares of the collocation equations. Solution of (8) is found from minimizing these functionals, with $\varPhi_1$ being minimized with respect to 10 first a_{jm} at fixed others, and $\varPhi_2$ being minimized at fixed a_{jm}, $j = 1, \ldots, 10$ with respect to the rest of a_{jm}.

Thus, we have the following system of equations for determining the coefficients a_{jm} in each cell:

$$\sum_{j=1}^{15} D_{lj} a_{jm} = \tilde{F}_l, \qquad l = 1, \ldots, 15. \tag{10}$$

It involves the same number of equations and unknowns.

The assemblage of these systems over all cells of the domain gives a global system of linear algebraic equations. A method of iterations over the subdomains is applied to solve this system. Each grid cell was treated as a subdomain. The solution is improved for each cell individually. At each iteration when improving the solution in the mth cell, solution in adjacent cells either already has been improved or was taken from previous iteration step. System (10) is solved by direct elimination method in each cell.

3 Application of CAS to Derive Formulas of the Method

To obtain formulas of CLS method we made programs in CAS REDUCE and Maple. These programs use the following elementary operations of CAS mentioned: arithmetic operations, functional dependencies, series summation, differentiation, substitutions, operations with common factors, simplifications, loops, operations with matrices. Governing equations, basic functions, form of the approximate solution, form of boundary and matching conditions, coordinates of collocation, matching and boundary nodes are the input parameters of the program. Quantities η, ξ, ζ, ω are used as symbols in order to obtain formulas in general form. For example, we introduce arrays **a**[ ] and **B**[ ] for coefficients a_j on current and previous iteration. Then we introduce functions **u(y1,y2), v(y1,y2),**

q(y1,y2) corresponding to $\widehat{u}_1$, $\widehat{u}_2$, $\widehat{q}$ and functions **uu(y1,y2), vv(y1,y2), qq(y1,y2)** corresponding to $\breve{u}_1$, $\breve{u}_2$, $\breve{q}$:

```
u(y1,y2):=sum('a[j]*fi1[j]','j'=1..15);
uu(y1,y2):=sum('B[j]*fi1[j]','j'=1..15);
v(y1,y2):=sum('a[j]*fi2[j]','j'=1..15);
vv(y1,y2):=sum('B[j]*fi2[j]','j'=1..15);
q(y1,y2):=sum('a[j]*fi3[j]','j'=1..15);
qq(y1,y2):=sum('B[j]*fi3[j]','j'=1..15);
```

Here **fi1**[**j**], **fi2**[**j**], **fi3**[**j**] stand for the first, second, and third component of basis vector-function φ_j. After filling in arrays **fi1**[**j**], **fi2**[**j**] and **fi3**[**j**] we write down the equations (5):

```
l1(y1,y2):=diff(u(y1,y2),y1$2)+diff(u(y1,y2),y2$2) -
  R*(diff(q(y1,y2),y1)+h*(uu(y1,y2)*diff(u(y1,y2),y1)+vv(y1,y2)
  *diff(u(y1,y2),y2)+u(y1,y2)*diff(uu(y1,y2),y1)+v(y1,y2)*diff(
  uu(y1,y2),y2)))-R*(h*h*f1(y1,y2)+diff(qq(y1,y2),y1)+h*(uu(y1,
  y2)*diff(uu(y1,y2),y1)+vv(y1,y2)*diff(uu(y1,y2),y2)))+diff(uu
  (y1,y2),y1$2)+diff(uu(y1,y2),y2$2);
l2(y1,y2):=diff(v(y1,y2),y1$2) + diff(v(y1,y2),y2$2) -
  R*(diff(q(y1,y2),y2)+h*(uu(y1,y2)*diff(v(y1,y2),y1)+vv(y1,y2)
  *diff(v(y1,y2),y2)+u(y1,y2)*diff(vv(y1,y2),y1)+v(y1,y2)*diff(
  vv(y1,y2),y2)))-R*(h*h*f2(y1,y2)+diff(qq(y1,y2),y2)+h*( uu(y1,
  y2)*diff(vv(y1,y2),y1)+vv(y1,y2)*diff(vv(y1,y2),y2)))+diff(vv
  (y1,y2),y1$2)+diff(vv(y1,y2),y2$2);
```

In order to specify the collocation equation we substitute coordinates of collocation node in l1(y1,y2) and l2(y1,y2) (W stands for ω):

```
eq_coll[1]:=subs(y1=-W,y2=-W,l1(y1,y2));
. . .
ek_coll[8]:=subs(y1=W,y2=W,l2(y1,y2));
```

Matching and boundary conditions are specified in the same manner.

We distinguish several types of cell: all sides of the cell are inside the domain ("inner cell"); one of the sides is on the domain boundary (4 types depending on position of this side); two adjacent sides are on the domain boundary (4 types too). Once all equations have been specified, the program collects functional of the sum of squared residuals of collocation equations and functionals of the sum of squared residuals of matching and boundary conditions for all mentioned cell types. The functional collected from the squared residuals of collocation equations does not depend on whether the cell side lies on the domain boundary or not and hence this functional is common for all cell types. The functional collected from squared residuals of matching and boundary conditions depends on the cell type. That is why we collect one functional corresponding to collocation equations and 9 functionals corresponding to matching and boundary conditions for different cell types. Collecting the "collocation" functional was made as follows:

```
Phi_coll:=sum('eq_coll[k]^2','k'=1..8);
```

Other functionals were collected analogously.

Then the program finds elements of matrices D and right-hand side vectors $\tilde{F}$ in (10) for all mentioned cell types by differentiating corresponding functionals with respect to unknown coefficients a_j and posing factors at a_j to corresponding places in matrices. Thus, for example, to obtain the 11th equation in (10) we should differentiate the functional Φ_2 with respect to the coefficient a_{11}:

```
row[11]:=diff(Phi_coll,a[11]);
```

and the element with indices i, j in the matrix D can be found by differentiating **row**[**i**] with respect to a_j:

```
for i from 1 to 15
do
 for j from 1 to 15 do
  D[i,j]:=diff(row[i],a[j]);
 od;
od;
```

In order to find the right-hand side $\tilde{F}$ in (10) we should substitute zeros in places of a_j into the expressions **-** **row**[**i**] (because **row**[**i**] corresponds to the equation $\frac{\partial \Phi_{1 \text{ or } 2}}{\partial a_i} = 0$):

```
for i from 1 to 15
do
FF[i]:=subs(a[1]=0,a[2]=0,a[3]=0,a[4]=0,a[5]=0,a[6]=0,
 a[7]=0,a[8]=0,a[9]=0,a[10]=0,a[11]=0,a[12]=0,a[13]=0,
 a[14]=0,a[15]=0, - row[i]);
od;
```

There are two possible ways to go further. The first one is to solve this system symbolically and to obtain symbolic formulas for every coefficient a_j in (6). This way is advantageous when a wide reduction is possible and formulas involve comparatively few number of arithmetic operations. The second way is to solve this system numerically already in the special program for finding numerical solution of the whole problem. It is also possible to combine rationally these two approaches.

The symbolic representations of elements in the matrices D and vectors $\tilde{F}$ which are obtained from the functionals corresponding to matching and boundary conditions are not very huge. But those elements that correspond to collocation equations are very huge. That is why we were unable to solve systems (10) symbolically and to obtain explicit expressions for a_j. So we have used the second approach mentioned above: the solution of this system is found numerically every time the improvement of approximate solution $(\breve{u}_1, \breve{u}_2, \breve{q})$ is sought for. For example, the representation for element $D_{11,11}$ occupies 5 rows, and expression for $\tilde{F}_{11}$ occupies 32 rows. Obviously, it is almost impossible for a researcher to derive these formulas with pen and paper without errors, and only computer algebra system can do this.

Moreover, it is possible to check the obtained formulas inside the CAS in which they were derived. Let us take some second order polynomials as an exact solution. Substituting it into equations (1) we obtain right-hand sides f_i of governing equations. Now, to check formulas for particular cell type we start calculating numerical solution only in the cell of interest taking right-hand sides of boundary and matching conditions from the exact solution. If the initial approximation $(\breve{u}_1, \breve{u}_2, \breve{q})$ in the cell is also the exact solution then the coefficients a_j of the improvement (6) of numerical solution will be equal to zero. If the initial approximation is equal to perturbed exact solution then the improvement of numerical solution will give exact solution in a few iterations. If such tests fail then it means that there are errors in the formulas.

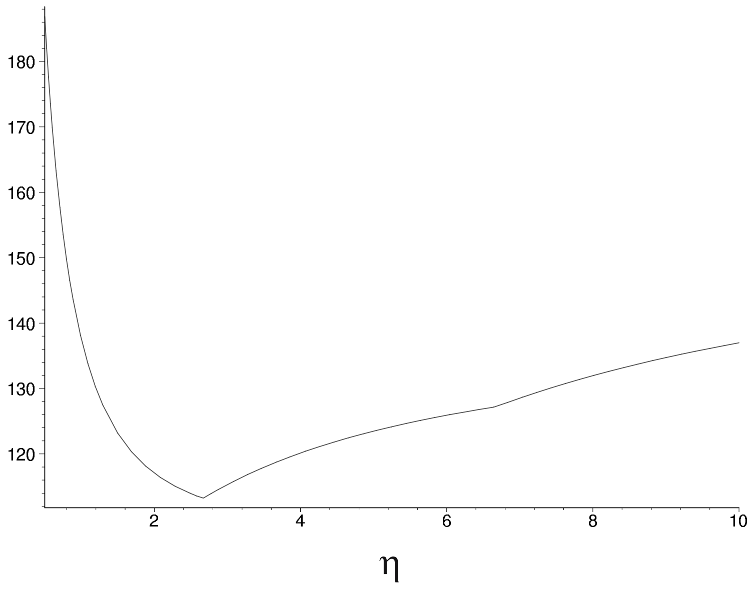


Fig. 1. The relationship between conditionality number and parameter η for the Stokes equations

We have also used CAS to analyse the condition number of the matrix D. It seems impossible to perform this analysis for the matrix of the method for Navier — Stokes equations because this matrix involves elements which are dependent on the numerical solution on the previous iteration and hence it changes from iteration to iteration. But in the case of Stokes equations the matrix D is independent of the numerical solution. It turned out to be possible for the **cond** operator to give symbolic formula for the condition number in this case. Then we used the **plot** operator to visualize the dependence of the condition number on the parameter η at different Reynolds numbers. Figure 1 presents this relationship at Reynolds number equaling 10. One can see that there is the value of η at which the condition number is minimal. It is well known that a large value of condition number affects the rate of convergence and the accuracy of computations and can cause the iteration method to be divergent. That is why it is very important to obtain SLAE with minimal possible condition number. Thus, CAS can help us to select the optimal value of the parameter of the method.

When the CAS program is verified on some variant of the method, it keeps the researcher from many possible errors at deriving formulas of new variants of the method. As a result, CAS allows efficient verification of the formulas of new method and facilitates and speeds up the formulas deriving procedure.

4 Some Numerical Results

This section presents the results of computations of test problems and model problems of a viscous flow in a cavity and flow over a backward-facing step. The computations of the test problems give information required for evaluation of the quality of discrete model. Subsequent solution of specific problems makes it possible to estimate the effectiveness and applicability of numerical algorithms developed.

For separated flows, including flows in a cavity and flow over a backward-facing step, special attention should be given to correct reproduction of a detailed flow pattern which is determined by the location of separation and reattachment points, sizes of the main and secondary vortices, etc. That is why a comparison of the results obtained in this study with those of other researches [7–10, 12–16] was made.

4.1 Test With Exact Solution Being Known

The validity of the formulas obtained was tested in the computational program on a number of problems with known exact solution beyond the basis. The following problem was used as one of the tests. The domain is a unity square. Trigonometric expressions for (v_1, v_2, p) were taken as exact solution: $v_1 = -\sin(\pi(x_2 - 0.5)/(1-2l))\cos(2\pi x_1)/(2(1-2l))$, $v_2 = \sin(2\pi x_1)\cos(\pi(x_2 - 0.5)/(1-2l))$, $p = (\cos(\pi/2x_1) + \cos(\pi/2x_2))/2$. We have assumed $l = 0.25$ in the numerical experiments. The substitution of this solution into (1) introduces terms of external forces and springs types into the right-hand sides, which are the causes of the liquid flow defined by this solution. Then, the CLS method was applied to find a numerical solution of the system (1) with right-hand sides obtained.

Under this test, numerical experiments were made on a sequence of grids to find the convergence order of the method. Differences between exact and numerical solutions were calculated on grids with step sizes 1/10, 1/20, 1/40 and 1/80. Comparative results of the velocity and pressure error computations on the grid sequence at the Reynolds number of 100 for variants of the CLS method with linear and quadratic pressure approximation for the test described above are presented in table 1.

Table 1. Defining the convergence order on grid sequence, Re=100

Grid step	Linear approximation of pressure		Quadratic approximation of pressure	
	Error	Error decrease factor	Error	Error decrease factor
Pressure				
1/10	9.5e-2		1.4e-1	
1/20	2.3e-2	4.1	8.9e-3	15.8
1/40	4.9e-3	4.7	1.4e-3	6.4
1/80	1.0e-3	4.9	2.4e-4	5.9
Velocity				
1/10	3.7e-2		6.0e-2	
1/20	8.9e-3	4.2	4.9e-3	12.3
1/40	2.3e-3	4.0	3.1e-4	15.8
1/80	5.1e-4	4.5	2.9e-5	10.7

One can see that the solution error improves by, at least, a factor of four when the grid step decreases by a factor of two both for the linear and the quadratic pressure approximation. Hence, the convergence order of the method is 2 or higher for smooth solutions.

Both the convergence order and the accuracy of pressure computing are improved in comparison with the case of pressure approximation with linear polynomials. It is worth noting that the convergence order and the accuracy of the velocity components are also improved. Thus, the improvement of the pressure approximation order leads to the improvement of the whole solution accuracy.

The fact that the numerical solution converges to the exact solution with the order expected is the additional evidence of correctness of formulas derived in CAS.

4.2 Flow Over Backward-Facing Step

The computation of the viscous liquid flow over a backward-facing step was performed using the proposed variant of CLS method with the pressure approximation by second-order polynomials. The step height was assumed to be equal to the half of the maximum domain height. A parabolic velocity profile was specified at the inlet, and another parabolic profile with the same discharge was specified at the outlet. The pressure was specified at the lower left corner of the inlet channel to determine it uniquely.

As pointed out in [7], the flow behind the step affects the flow behaviour in the inlet channel, namely the parabolic profile of the flow is distorted near the

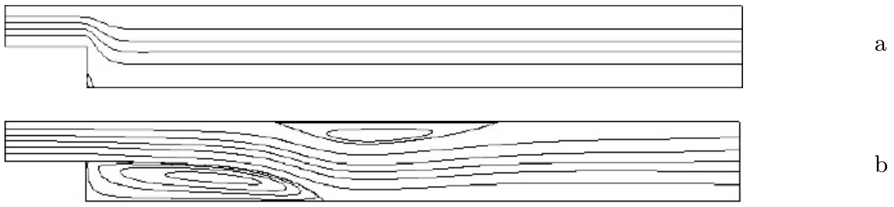


Fig. 2. Flow over step, Re = 1 (a), Re = 800 (b)

sudden expansion point. That is why the length of the inlet channel should be set essentially nonzero.

The computations were made in the Reynolds number range from 1 to 800. It is known [7] that the flow is laminar within this range.

Figure 2 shows the streamlines at different Reynolds numbers. The streamlines were reconstructed from the values of the velocity components obtained by the CLS method. The scale of ordinate axis was taken two times larger than the scale of abscissa axis for better visualization. Obviously, recirculation regions behind the step appear and expand with increasing Reynolds number.

The validity of the qualitative structure of the flow is confirmed by physical experiments. The coordinates of the separation and reattachment points are distinctive quantitative values that define the flow structure. At low Reynolds numbers there is good agreement in these coordinates obtained by CLS method and by numerical results of other researches. At Re = 800 the first recirculation region in experiment and 3D computations is longer than in 2D computations, including the computations performed in the present study. Apparently, this effect is caused by an essential three - dimensionality of this flow at large Reynolds numbers. As noted in [8], a good coincidence of the 2D results and the experimental data is observed at Reynolds numbers up to 500.

4.3 Flow in the Lid-Driven Cavity

The computations of the flow in a cavity with driven lid were performed for the Reynolds numbers from Re=1 to Re=2000 for different depth-to-width ratios. The streamline patterns for Reynolds numbers 1 and 1000 in the square cavity are shown in Figure 3. The vortex center is marked by the '+' sign. An enlargement of the bottom corner vortices and a shift of the main vortex center initially along the lid motion direction and, then, to the cavity center are seen to occur when the Reynolds number increases. This effect is confirmed by physical experiments [11] and computations [12,13,14,15,16].

A comparison of the maximum and minimum velocity values along the vertical and horizontal cavity centerlines with the results published in [13,14,15,16] shows that distinctive values of the velocity profile obtained by the CLS method agree well with those obtained by other researchers.

It is worth to note that in our computations the corner vortices are present even at Reynolds number 1, although they are very weak.

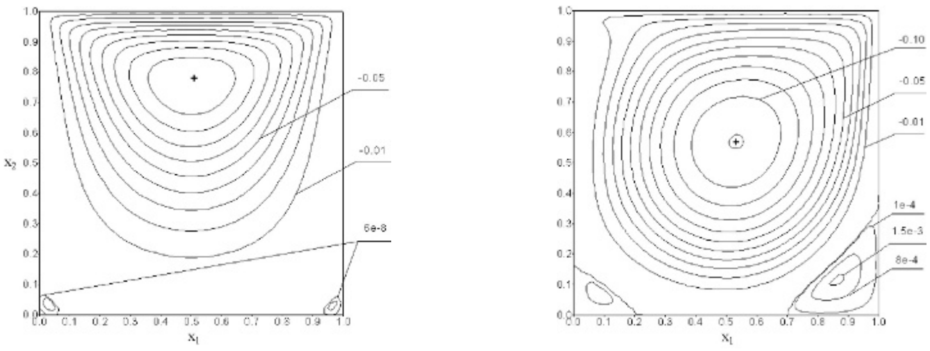


Fig. 3. Streamlines of the flow in the cavity. Re = 1 (left), Re = 1000 (right).

5 Summary

In the present study we tried to show the very important role of CAS for constructing a numerical method for solving equations of mathematical physics. The CAS program for constructing the formulas of the method is flexible and allows to construct other modifications of the method. The usage of CAS keeps the researcher from many possible arithmetical errors and allows one to perform symbolic manipulations with very huge analytical expressions. Computer derivation of the formulas also gives additional advantages. Optimization with respect to the rate of computation by obtained formulas, analytical and numerical verification of these formulas by using the same or another CAS, conversion of constructed formulas to arithmetic operators of FORTRAN and C programming languages can be made. All these factors greatly facilitate the process and reduce the time for reaching final result.

Hence, the advantage of the computer algebra application for construction and analysis of the formulas for a numerical method is obvious.

References

1. Semin, L.G., Sleptsov, A.G., Shapeev, V.P.: Collocation and least - squares method for Stokes equations. Computational technologies **1** No. 2 (1996) 90–98 (in Russian)
2. Valiullin, A.N., Shapeev, V.P. et al.: Symbolic manipulations in the methods of mathematical physics. Symposium: Mathematics for Computer Science. Paris, March 16–18 (1982), 431–438
3. Steinberg, S.: A problem solving environment for numerical partial differential equations. 6th IMACS Conference on Application of Computer Algebra, St. Petersburg, June 25–28 (2000), 98–99
4. Karasözen, B., Tsybulin, V.G.: Conservative finite difference schemes for cosymmetric systems. Computer Algebra in Scientific Computing, Springer, Berlin (2001) 363–375

5. Schacht, W., Vorozhtsov, E.V.: Implementation of Roe's method for numerical solution of three-dimensional fluid flow problems with the aid of computer algebra systems. 7th workshop on Computer Algebra in Scientific Computing, St. Petersburg, July 12–19 (2004), 409–422
6. Ascher, U., Christiansen, J., Russell, R.D.: A collocation solver for mixed order systems of boundary value problems. Mathematics of Computation **33** (1979) 659–679
7. Barton, I.E.: The entrance effect of laminar flow over a backward-facing step geometry. Int. J. for Numerical Methods in Fluids **25** (1997) 633–644
8. Kim, J., Moin, P.: Application of a fractional-step method to incompressible Navier — Stokes equations. J. Comput. Phys. **59** (1985) 308–323
9. Armaly, B.F., Durst, F., Pereira, J.C.F., Schönung, B.: Experimental and theoretical investigation of backward-facing step flow. J. Fluid Mech. **127** (1983) 473–496
10. Oosterlee, C.W.: A GMRES-based plane smoother in multigrid to solve 3D anisotropic fluid flow problems. J. Comput. Phys. **130** (1997) 41–53
11. Pan, F., Acrivos, A.: Steady flows in rectangular cavities. J. Fluid Mech. **28** pt. 4 (1967) 643–655
12. Bozeman, J.D., Dalton, C.: Numerical study of viscous flow in a cavity. J. of Comput. Phys. **12** (1973) 348–363
13. Bruneau, C.H., Jouron, C.: An efficient scheme for solving steady incompressible Navier — Stokes equations. J. Comput. Phys. **89** (1990) 389–413
14. Ghia, U., Ghia, K.N., Shin, C.T.: High – Re solutions for incompressible flow using the Navier — Stokes equations and a multigrid method. J. Comput. Phys. **48** (1982) 387–411
15. Deng, G.B., Piquet, J., Queutey, P., Visonneau, M.: A new fully coupled solution of the Navier — Stokes equations. Int. J. Numer. Methods in Fluids **19** (1994) 605–639
16. Chen, C.J., Chen, H.J.: Finite analytic numerical method for unsteady two-dimensional Navier — Stokes equations. J. Comput. Phys. **53** (1984) 209–226

Nonlinear Waves in a Rod

Alexander Shermenev

Wave Research Center, Russian Academy of Sciences, Moscow, 117942
sher@orc.ru

Abstract. Computer algebra system is applied for studying the elastic torsional nonlinear waves in a rod using the second order approximation. It is shown that the nonlinear correction to the classic linear solution is a combination of a stationary longitudinal wave, a progressive longitudinal wave, and a progressive transverse wave. The solution describing a stationary longitudinal wave is a quadratic polynomial of cylindrical functions. The expressions for a progressive longitudinal wave and a progressive transverse wave inevitably include quadratures from polynomials of the cylindrical functions.

1 Introduction

Elastic torsional linear waves in a rod are studied in almost all classic books on elastic waves in solids and give a typical example of transverse waves (S-waves) (see [1]–[3]). The purpose of this work is to calculate the nonlinear quadratic correction for the torsional waves. This correction is naturally decomposed in three terms: the stationary longitudinal wave, the progressive longitudinal wave, and the progressive transverse wave.

The main result concerns the stationary longitudinal waves. Substituting the expression

$$u_\theta = aJ_1(kr)\sin(Kz - \omega t), \quad u_r = 0, \quad u_z = 0, \tag{1}$$

which describes the torsional waves in the linear approximation, in the nonlinear terms of elasticity equation, we obtain a quadratic polynomial in Bessel functions $J_0(kr)$ and $J_1(kr)$ whose coefficients are rational functions of r. When we search for a nonlinear correction in the same form

$$R_{00}J_0^2 + R_{01}J_0J_1 + R_{11}J_1^2, \tag{2}$$

where R_{AB} are polynomials of r^{-1} and r with unknown coefficients and of unknown degree n

$$R_{AB} = \sum_{k=-n}^{n} C_k^{AB} r^k. \tag{3}$$

we substitute (2) in linear part of the elasticity equation and obtain **overdetermined** systems of linear algebraic equations for C_k^{AB}. The key point of consideration is that these **overdetermined** systems are compatible for the case of the stationary longitudinal waves. This allows us to construct explicit expressions for the nonlinear correction, which are homogenous polynomials of the Bessel functions $J_0(kr)$ and $J_1(kr)$. The result can be interpreted as integrability of some cubic expressions of Bessel functions. The situation is similar to the case of waves in the fluid and gas where the nonlinear solution always is a quadratic polynomial of the cylindrical functions. (Classic examples are studied in [4]–[7], and the general result is proved in [8]). Presence of a standing wave means that the rod with torsional waves is subjected to a constant radial deformation.

The expressions describing the progressive longitudinal wave and the progressive transverse wave include a solution of a non-homogenous ordinary differential second-order equation of the form

$$Y_{rr} + \frac{1}{r}Y_r + \gamma Y + J(r)^2 = 0 \tag{4}$$

which can be expressed in terms of quadratures from cubic polynomials of Bessel functions. Functions of this type appear also in other nonlinear problems and deserve detailed study which is started in this paper. We are focused on searching for particular solutions of equations (4) assuming that remaining solutions can be obtained by addition of Bessel functions. Analysis of the second-order approximations in nonlinear mathematical physics demands a considerable amount of symbolic calculations and hardly can be performed without computer algebra system. Mathematica 4.1 was used for writing this article.

2 Nonlinear Theory of Elasticity

Here we shall remind basic formulas of the three-dimensional nonlinear elasticity theory (See [1]). The elastic energy of an isotropic solid in the third approximation can be expressed in terms of a vector function of deformation $\mathbf{u}(x_1, x_2, x_3)$ as follows:

$$\begin{aligned}\mathcal{E} = {} & \tfrac{\mu}{4}\left(\tfrac{\partial u_i}{\partial x_k} + \tfrac{\partial u_k}{\partial x_i}\right)^2 + \tfrac{\lambda}{2}\left(\tfrac{\partial u_l}{\partial x_l}\right)^2 \\ & + \left(\mu + \tfrac{A}{4}\right)\tfrac{\partial u_i}{\partial x_k}\tfrac{\partial u_l}{\partial x_i}\tfrac{\partial u_l}{\partial x_k} + \left(\tfrac{B}{2} + \tfrac{\lambda}{2}\right)\tfrac{\partial u_l}{\partial x_l}\left(\tfrac{\partial u_i}{\partial x_k}\right)^2 \\ & + \tfrac{A}{12}\tfrac{\partial u_i}{\partial x_k}\tfrac{\partial u_k}{\partial x_l}\tfrac{\partial u_l}{\partial x_i} + \tfrac{B}{2}\tfrac{\partial u_i}{\partial x_k}\tfrac{\partial u_k}{\partial x_i}\tfrac{\partial u_l}{\partial x_l} + \tfrac{C}{3}\left(\tfrac{\partial u_l}{\partial x_l}\right)^3 .\end{aligned} \tag{5}$$

Calculating a stress tensor

$$\sigma_{ij} = \frac{\partial \mathcal{E}}{\partial\left(\partial u_i/\partial x_k\right)}, \tag{6}$$

we obtain

$$\begin{aligned}\sigma_{ik} = {} & \mu\left(\tfrac{\partial u_i}{\partial x_k} + \tfrac{\partial u_k}{\partial x_i}\right) + \lambda\left(\tfrac{\partial u_l}{\partial x_l}\right)\delta_{ik} \\ & + \left(\mu + \tfrac{A}{4}\right)\left(\tfrac{\partial u_l}{\partial x_i}\tfrac{\partial u_l}{\partial x_k} + \tfrac{\partial u_i}{\partial x_l}\tfrac{\partial u_k}{\partial x_l} + \tfrac{\partial u_i}{\partial x_l}\tfrac{\partial u_l}{\partial x_k}\right) \\ & + \left(\tfrac{\lambda}{2} + \tfrac{B}{2}\right)\left(\left(\tfrac{\partial u_l}{\partial x_m}\right)^2\delta_{ik} + 2\tfrac{\partial u_i}{\partial x_k}\tfrac{\partial u_l}{\partial x_l}\right) \\ & + \tfrac{A}{4}\tfrac{\partial u_k}{\partial x_l}\tfrac{\partial u_l}{\partial x_i} + \tfrac{B}{2}\left(\tfrac{\partial u_l}{\partial x_m}\tfrac{\partial u_m}{\partial x_l}\delta_{ik} + 2\tfrac{\partial u_k}{\partial x_i}\tfrac{\partial u_l}{\partial x_l}\right) + C\left(\tfrac{\partial u_l}{\partial x_l}\right)^2\delta_{ik} .\end{aligned} \tag{7}$$

Thus, the nonlinear elasticity equation has the following form:

$$\frac{\partial^2 u_i}{\partial t^2} - \mu \frac{\partial^2 u_i}{\partial x_k^2} - (\mu + \lambda) \frac{\partial^2 u_l}{\partial x_l \partial x_i} = F_i \tag{8}$$

where

$$\begin{aligned} F_i = {} & \left(\mu + \tfrac{A}{4}\right)\left(\tfrac{\partial^2 u_l}{\partial x_k^2}\tfrac{\partial u_l}{\partial x_i} + \tfrac{\partial^2 u_l}{\partial x_k^2}\tfrac{\partial u_i}{\partial x_l} + 2\tfrac{\partial^2 u_i}{\partial x_l \partial x_k}\tfrac{\partial u_l}{\partial x_k}\right) \\ & + \left(\mu + \tfrac{A}{4} + \lambda + B\right)\left(\tfrac{\partial^2 u_l}{\partial x_i \partial x_k}\tfrac{\partial u_l}{\partial x_k} + \tfrac{\partial^2 u_k}{\partial x_l \partial x_k}\tfrac{\partial u_i}{\partial x_l}\right) + (\lambda + B)\left(\tfrac{\partial^2 u_i}{\partial x_k^2}\tfrac{\partial u_l}{\partial x_l}\right) \\ & + \left(\tfrac{A}{4} + B\right)\left(\tfrac{\partial^2 u_k}{\partial x_l \partial x_k}\tfrac{\partial u_l}{\partial x_i} + \tfrac{\partial^2 u_l}{\partial x_i \partial x_k}\tfrac{\partial u_k}{\partial x_l}\right) + (B + 2C)\left(\tfrac{\partial^2 u_k}{\partial x_i \partial x_k}\tfrac{\partial u_l}{\partial x_l}\right). \end{aligned} \tag{9}$$

We assume that

$$\mathbf{u} = \varepsilon \text{ grad } \varphi + \varepsilon^2 \text{ grad } \varphi'' + \varepsilon \text{ rot } \psi + \varepsilon^2 \text{ rot } \psi'', \tag{10}$$

and try to find analogous decomposition of equation (8).

3 Special Case

We consider solutions which have the following form in the linear approximation

$$u_1 = \partial_{x_2} U, \quad u_2 = -\partial_{x_1} U, \quad u_3 = 0, \tag{11}$$

where

$$U = \varepsilon X(x_1, x_2) \sin (K x_3 - \omega t). \tag{12}$$

A deformation $\mathbf{u}$ satisfies the linear version of the elasticity equation (8) if and only if the function $X(x_1, x_2)$ is a solution of the Helmholtz equation

$$X_{x_1 x_1} + X_{x_2 x_2} + \frac{\omega^2 - K^2 \mu}{\mu} X = 0 \tag{13}$$

or differs from its solution by constant.

If

$$X(x_1, x_2) = X\left(\sqrt{x_1^2 + x_2^2} = r\right), \tag{14}$$

we obtain a torsional wave in a rod. Substituting (14) in (13), we have the following equation for $X(r)$

$$X'' + \frac{1}{r} X' + \frac{\omega^2 - \mu K^2}{\mu} X = 0. \tag{15}$$

Its solution regular at zero is the Bessel function

$$X(r) = J_0(kr), \tag{16}$$

where $k = \frac{\omega^2 - \mu K^2}{\mu}$. Then we have

$$u_\theta = a J_1(kr) \sin (Kz - \omega t), \quad u_r = 0, \quad u_z = 0. \tag{17}$$

4 Calculation of Nonlinear Terms

Substituting (12) in (9), we have

$$\begin{aligned} F_1 &= \widetilde{C}_1(x_1,x_2) + C_1(x_1,x_2)\cos(2Kz-2\omega t) \\ F_2 &= \widetilde{C}_2(x_1,x_2) + C_2(x_1,x_2)\cos(2Kz-2\omega t) \\ F_3 &= \widetilde{C}_3(x_1,x_2) + C_3(x_1,x_2)\sin(2Kz-2\omega t) \end{aligned} \tag{18}$$

where $\widetilde{C}_i(x_1,x_2)$ and $C_i(x_1,x_2)$ are linear combinations of

$$XX_{x_1}, X_{x_2}X_{x_1x_2}, X_{x_1}X_{x_1x_1}, X_{x_1x_2}X_{x_1x_1x_2}, XX_{x_1x_1x_1}, X_{x_1x_1}X_{x_1x_1x_1} \tag{19}$$

with numerical coefficients (these coefficients can be explicitly calculated in terms of μ, λ, A, B, K, and ω. They do not depend on C).

We would like to represent F_i in the form

$$F = \text{grad }\varphi^F + \text{curl }\psi^F \tag{20}$$

Potential $\varphi^F(x_1,x_2,x_3)$ can be found using the following condition

$$\partial_{x_1}\left(F_1 - \varphi^F_{x_1}\right) + \partial_{x_2}\left(F_2 - \varphi^F_{x_2}\right) + \partial_{x_3}\left(F_3 - \varphi^F_{x_3}\right) = 0 \tag{21}$$

Naturally, we have

$$\varphi^F = \varphi^F_0 + \varphi^F_2\cos(2Kz-2\omega t) \tag{22}$$

Assume that a function $Q(x_1,x_2)$ satisfies the condition

$$Q_{x_1x_1} + Q_{x_2x_2} - 4K^2Q + X^2 = 0 \tag{23}$$

Then computer algebra calculations give the expressions

$$\begin{aligned} \varphi^F_0 &= \tfrac{1}{16\mu^2}\left(K^2\mu-\omega^2\right)\left(K^2\mu\,(A+4B+4\lambda+4\mu)\right.\\ &\quad \left.-\,2\,(A+2B+2\lambda+4\mu)\,\omega^2\right)X^2\\ &\quad + \tfrac{1}{16}K^2\,(A+4B+4\lambda+4\mu)\left(X^2_{x_1}+X^2_{x_2}\right)\\ &\quad + \tfrac{1}{2\mu}\left(K^2\mu-\omega^2\right)(A+2B+\lambda+3\mu)\,XX_{x_1x_1}\\ &\quad + \tfrac{1}{2}(A+2B+\lambda+3\mu)\left(X^2_{x_1x_1}+X^2_{x_1x_2}\right);\\ \varphi^F_2 &= \tfrac{K^2(A+4\mu)\omega^4}{2\mu^2}Q(x_1,x_2) - \tfrac{1}{16\mu^2}\left(K^4\mu^2\,(A+4B+4\lambda+4\mu)\right.\\ &\quad \left.-\,K^2\mu\,(A+8B+8\lambda+4\mu)\,\omega^2 + 2\,(A+2B+2\lambda+4\mu)\,\omega^4\right)X^2\\ &\quad + \tfrac{K^2}{16}(A+4B+4\lambda+4\mu)\left(X^2_{x_1}+X^2_{x_2}\right)\\ &\quad + \tfrac{1}{2}\left(K^2-\tfrac{\omega^2}{\mu}\right)(A+2B+\lambda+3\mu)\,XX_{x_1x_1}\\ &\quad - \tfrac{1}{2}(A+2B+\lambda+3\mu)\left(X^2_{x_1x_2}+X^2_{x_1x_1}\right), \end{aligned} \tag{24}$$

which determine a potential φ^F satisfying (21).

Without loss of generality, we can assume that vector potential ψ^F has the following form

$$\psi^F = \left(\psi^F_1\,(x_1,x_2)\sin(2Kx_3-2\omega t)\,,\psi^F_2\,(x_1,x_2)\sin(2Kx_3-2\omega t)\,,0\right). \tag{25}$$

Therefore, we have

$$\begin{aligned} F_1 - \varphi^F_{x_1} &= 2K\psi^F_2 \cos\left(2Kx_3 - 2\omega t\right), \\ F_2 - \varphi^F_{x_2} &= -2K\psi^F_1 \cos\left(2Kx_3 - 2\omega t\right), \\ F_3 - \varphi^F_{x_3} &= \left(\psi^F_{2x_1} - \psi^F_{1x_2}\right) \sin\left(2Kx_3 - 2\omega t\right). \end{aligned} \tag{26}$$

Hence

$$\begin{aligned} \psi^F_1 &= \tfrac{K(A+4\mu)}{4} \left(-\tfrac{\omega^4}{\mu^2} Q_{x_2} + \tfrac{\omega^2}{2\mu} X X_{x_2} + X_{x_1} X_{x_1 x_2} - X_{x_2} X_{x_1 x_1}\right), \\ \psi^F_2 &= \tfrac{K(A+4\mu)}{4} \left(\tfrac{\omega^4}{\mu^2} Q_{x_1} + \tfrac{\omega^2\left(2K^2\mu - 3\omega^2\right)}{2\mu} X X_{x_1} - X_{x_2} X_{x_1 x_2} - X_{x_1} X_{x_1 x_1}\right). \end{aligned} \tag{27}$$

5 *P*−Waves

A potential φ'' from expression (10) satisfies the equation

$$(\lambda + 2\mu)\left(\varphi''_{x_1x_1} + \varphi''_{x_2x_2} + \varphi''_{x_3x_3}\right) - \varphi''_{tt} + \varphi^F = 0 \tag{28}$$

We assume that $\varphi'' = f_0(x_1, x_2) + f_2(x_1, x_2)\cos(2Kx_3 - 2\omega t)$ and have the following equations for f_0 and f_2:

$$(\lambda + 2\mu)\left(f_{0x_1x_1} + f_{0x_2x_2}\right) + \varphi^F_0 = 0 \tag{29}$$

$$(\lambda + 2\mu)\left(f_{2x_1x_1} + f_{2x_2x_2} - 4K^2 f_2\right) + \varphi^F_2 = 0 \tag{30}$$

Stationary Solution. Let $Q_0(x_1, x_2)$ be a solution of equation

$$Y_{x_1x_1} + Y_{x_2x_2} + X^2 = 0 \tag{31}$$

Then it can be checked that the expression

$$\begin{aligned} S = {} & \tfrac{\left(-3A\mu - 4B\mu - 8\mu^2\right)K^2 + (4A + 8B + 4\lambda + 12\mu)\omega^2}{32\mu(\lambda + 2\mu)} X^2 \\ & + \tfrac{A + 2B + \lambda + 3\mu}{8(\lambda + 2\mu)} \left(X^2_{x_1} + X^2_{x_2}\right) \\ & + \tfrac{\left(-3A\mu - 4B\mu - 8\mu^2\right)K^2 + (4A + 8B + 4\lambda + 12\mu)\omega^2}{32\mu(\lambda + 2\mu)} Q_0 \end{aligned} \tag{32}$$

gives a particular solution of equation (29).

Assume that all considered functions depend only on $r = \sqrt{x_1^2 + x_2^2}$ and denote $J_0(kr)$ by J. Then equation (31) takes the form

$$Y_{rr} + \frac{1}{r} Y_r + J^2 = 0 \tag{33}$$

and the expression

$$Q^{Polar}_0 = -\frac{1}{2}\, r^2 J^2 + \frac{r}{2\left(K^2 - \frac{\omega^2}{\mu}\right)} J J' + \frac{r^2}{2\left(K^2 - \frac{\omega^2}{\mu}\right)} J'^2 \tag{34}$$

is its particular solution.

Expression for S takes the form

$$\begin{aligned} S^{Polar}(r) = &\frac{\left(-3A\mu-4B\mu-8\mu^2\right)K^2+(4A+8B+4\lambda+12\mu)\omega^2}{32\mu(\lambda+2\mu)}J^2 \\ &+\frac{A+2B+\lambda+3\mu}{8(\lambda+2\mu)}J'^2 \\ &+\frac{\left(-3A\mu-4B\mu-8\mu^2\right)K^2+(4A+8B+4\lambda+12\mu)\omega^2}{32\mu(\lambda+2\mu)}Q_0^{Polar} \end{aligned} \tag{35}$$

Expression for Deformations:

$$u_1 = S^{Polar}_{x_1}, \quad u_2 = S^{Polar}_{x_2}, \quad u_r = S^{Polar\prime}, \quad u_\theta = 0. \tag{36}$$

Periodic Solution. Let $Q_2(x_1, x_2)$ be a solution of equation

$$Y_{x_1x_1} + Y_{x_2x_2} + \frac{4\left(\omega^2 - K^2(\lambda+2\mu)\right)}{8(\lambda+2\mu)}Y + X^2 = 0. \tag{37}$$

Then it can be checked that the expression

$$\begin{aligned} &\frac{(3A+4B+8\mu)K^2\mu(\lambda+2\mu)+4\lambda(A+2B+\lambda+3\mu)\omega^2}{32\mu(\lambda+2\mu)^2}X^2 \\ &+\frac{A+2B+\lambda+3\mu}{8(\lambda+2\mu)}\left(X_{x_1}^2+X_{x_2}^2\right)+\frac{(A+4\mu)K^2\omega^2}{8\mu^2}Q_2 \end{aligned} \tag{38}$$

gives a particular solution of equation (30). Assume that all considered functions depend only on $r = \sqrt{x_1^2 + x_2^2}$. Then equation (37) takes the form

$$Y_{rr} + \frac{1}{r}Y_r + \frac{4\left(\omega^2 - K^2(\lambda+2\mu)\right)}{8(\lambda+2\mu)}Y + J^2 = 0 \tag{39}$$

and, therefore, $Q_2(r)$ must be a particular solution of this Bessel-type inhomogeneous equation.

6 Numerical Example, Stationary $P-$Waves

Here we present an example of stationary P-waves for the following values of parameters:

$\lambda = 11.6$, $\mu = 8.4$, $A = 11$, $B = -15.8$, $K = 17.2$, $\omega = 2$.

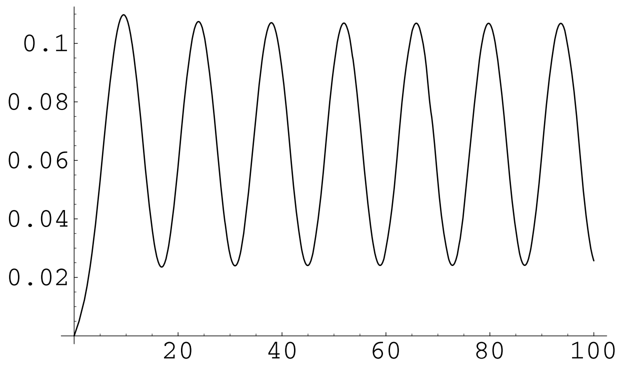


Fig. 1. Radial deformation u_r versus radius r

7 $S-$Waves

Equation for ψ'' has the form

$$\mu\left(\psi''_{x_1x_1}+\psi''_{x_2x_2}+\psi''_{x_3x_3}\right)-\psi''_{tt}+\psi^F=0 \tag{40}$$

We assume that

$$\psi''=\left(a_1\left(x_1,x_2\right)\sin\left(2Kx_3-2\omega t\right),a_2\left(x_1,x_2\right)\sin\left(2Kx_3-2\omega t\right),0\right)$$

and have the following equations for $a_1\left(x_1,x_2\right)$ and $a_2\left(x_1,x_2\right)$:

$$\begin{array}{l}\mu\left(a_{1x_1x_1}+a_{1x_2x_2}\right)-4\left(K^2\mu-\omega^2\right)a_1+\psi_1=0\\ \mu\left(a_{2x_1x_1}+a_{2x_2x_2}\right)-4\left(K^2\mu-\omega^2\right)a_2+\psi_2=0\end{array} \tag{41}$$

Then it can be checked that the expressions

$$\begin{array}{l}a_1^*\left(x_1,x_2\right)=\frac{(A+4\mu)K}{8\mu}XX_{x_2}-\frac{(A+4\mu)K\omega^2}{16\mu^2}Q_{x_2},\\ a_2^*\left(x_1,x_2\right)=-\frac{(A+4\mu)K}{8\mu}XX_{x_1}+\frac{(A+4\mu)K\omega^2}{16\mu^2}Q_{x_1}\end{array} \tag{42}$$

give particular solutions of equations (41). Assume that all considered functions depend only on $r=\sqrt{x_1^2+x_2^2}$. Then equation (23) for $Q\left(r\right)$ takes the form

$$Y_{rr}+\frac{1}{r}Y_r+K^2Y+J^2=0 \tag{43}$$

and, therefore, $Q\left(r\right)$ must be a particular solution of the Bessel type equation.

8 Conclusions

We have constructed a solution of the nonlinear equation of the elasticity theory whose linearization gives classic torsional waves in a rod with circular section. The constructed solution is decomposed into a stationary (standing) longitudinal wave, a progressive longitudinal wave, and a progressive transverse wave. The standing wave is represented by explicit quadratic expression of cylindrical functions. Its presence means a stationary radial deformation of a rod with torsional waves. The expressions for progressive waves include quadratures from cubic polynomials of cylindrical functions.

References

1. Landau, L.D., Lifshits, E.M.: Theory of Elasticity. Third Ed., Pergamon Press, Oxford (1986)
2. Love, A.E.: A Treatise on the Mathematical Theory of Elasticity. 4th Ed., New York (1944)
3. Rayleigh, W.: Theory of Sound, McMillan, London (1929)
4. Shermenev, A., Shermeneva, M.: Long periodic waves on an even beach. Physical Review E, No. 5 (2000) 6000–6002

5. Shermenev, A.: Nonlinear periodic waves in shallow water. In: Proc. Conf. for Symbolic and Numerical Scientific Computation, LNCS **2630**, Springer-Verlag (2002) 375–386
6. Shermenev, A.: Nonlinear acoustical waves in tubes. Acta Acoustica **89** (2003) 426–429
7. Shermenev, A.: Wave equation in special coordinates. J. Nonlinear Math. Physics 11 (2004) (Supplement) 110–115
8. Shermenev, A.: Separation of variables for nonlinear wave equation in polar coordinates. J. Physics, A **37** (2004) 1–9

A Symbolic-Numeric Method for Solving Boundary Value Problems of Kirchhoff Rods

Liu Shu and Andreas Weber

Institut für Informatik II, Universität Bonn,
Römerstr. 164, Bonn, Germany
{liushu, weber}@cs.uni-bonn.de

Abstract. We study solution methods for boundary value problems associated with the static Kirchhoff rod equations. Using the well known Kirchhoff kinetic analogy between the equations describing the spinning top in a gravity field and spatial rods, the static Kirchhoff rod equations can be fully integrated. We first give an explicit form of a general solution of the static Kirchhoff equations in parametric form that is easy to use. Then by combining the explicit solution with a minimization scheme, we develop a unified method to match the parameters and integration constants needed by the explicit solutions and given boundary conditions. The method presented in the paper can be adapted to a variety of boundary conditions. We detail our method on two commonly used boundary conditions.

1 Introduction

The study of deformations in elastic rods can be applied in several fields. Examples of slender structures in structural engineering are submarine cables or tower cables [1]; in biology elastic rods are often used to study supercoiling and other mechanical behaviors of DNA strands [2,3,4,5]. And in daily life we often see string objects or filaments, like telephone cords, willow branches, climbing plants and human or animals hair [6].

Based on Newton's second law, the Kirchhoff rod model provides a theoretical frame describing the static and dynamic behaviors of elastic rods [7,8]. The Kirchhoff model holds for small curvatures of rods, but Kirchhoff rods can undergo large changes of shape [9]. In particular, for the static case, all dependent variables appearing in the equations are only functions of one spatial variable, such as arc length of rods. Thus the static Kirchhoff equations are a set of ordinary differential equations. Numeric method can be used to solve the associated initial value problems (IVP) or boundary value problems (BVP), but the Kirchhoff equations are well known to be difficult for numeric methods due to their stiffness [10].

On the other side, a well known feature of the Kirchhoff rod model is called Kirchhoff kinetic analogy. Theoretically, the governing equations of the static Kirchhoff rods are formally equivalent to the Euler equations describing the motion of a rigid body with a fixed point under external force fields [7]. In some instances, the Euler equations are fully integrable [7], e.g. in the famous case investigated by Sofia Kovalevskaya [11]. In the case of rods, the corresponding example is the Kovalevskaya rod [12], the governing equations of which are fully integrable by the analogy. So far, several achievements have been made to obtain the symbolic solutions to the Kirchhoff equations. Shi and Hearst first obtained an explicit form of solution of the static Kirchhoff equations [13]. Nizette and Goriely gave a parameterized analytical solution for Kirchhoff rods with circular cross-section and further made a systematic classification of all kinds of equilibrium solutions [7]. Goriely et al. studied the dynamical stability of elastic strips by analyzing the amplitude equations governing the dynamics of elastic strips [8].

Unfortunately the above achievements can not be easily used in real applications, because in real situations, generally we deal with finite rods constrained by specific boundary conditions. Pai presented a two-phase integration method to model the behaviors of the strand of surgical suture [14]. The full static Kirchhoff equations including distributed external loading and initial curvatures of rods are considered. Because of its fast computational speed it matches the request of operating rods in real time for computer graphics. However the scheme is not complete in theory. It may be used to determine the shape of rods in the case where the final shape of rods has only small changes compared to the initial shape of rods. In case of large changes in shape of rods, two steps of integration may not result in a good precision. So one needs an iterative procedure, in which the two-step integration will be repeated until a given precision is matched. However, we do not know of any formal proof that guarantees the convergence of the approach. Combining the shooting method and monodromy method, da Fonseca et al. in their study of the equilibrium solution of DNA gave a method for solving a boundary value problem associated with the Kirchhoff rods [9]. As is pointed out in [9], the scheme may fail if the end point of rods is moved into a forbidden area.

1.1 Our Contribution

In this paper we develop a symbolic-numeric method for solving boundary value problems associated with the static Kirchhoff rods, which works uniformly for various BVPs. A major difference between our work and that presented in [9] is that our method is motivated in the application of physics-based human hair modeling. We use Kirchhoff rods to model hair fibers. It is possible that different kinds of boundary conditions can come up. For example, there are at least two kinds of boundary conditions commonly used in hair modeling. For the first case of interest which was also dealt with in Pai's model [14], the considered rod is clamped at one end point and at the other end of rods external forces and torques are exerted. Another possibility arising in applications is that the positions of both ends of rods are given and at one end point of rods the orientation is given, too. In [9] a similar boundary condition was dealt with, where only the positions of both ends of rods were given at boundaries of rods. In our work we first study the parametric closed form of solution to the static Kirchhoff equations and express it in a form easy to be used for matching user given boundary conditions. Then we provide a method matching the given boundary conditions by solving an unconstrained global minimization problem, the solution of which is the set of parameters solving the parametric explicit form of the solution of the static Kirchhoff equations. Our method is a general method for solving BVPs, which can be adapted to a variety of boundary conditions. We will give detailed results of our method being applied to the two boundary conditions mentioned above.

2 Closed Form Solution of the Static Kirchhoff Equations

2.1 Geometric Representation of 3D Rods

A curly rod can approximately be represented by a spatial curve because of its special geometric feature. The deformations of any point on the cross-section of rods have little contribution to the final shape of rods subjected to external loads. Thus the axis of rods, or the corresponding spatial curve of rods, can be parameterized by arc length. At any time, for every point on the axis of rods, say s, we have a position vector $\mathbf{R}(s)$ and three orthonormal vectors $\mathbf{d}_1(s), \mathbf{d}_2(s), \mathbf{d}_3(s)$ constructing a local triad there. Without loss of generality we may assume that the vector $\mathbf{d}_3(s)$ is the tangent of the rod at point s, while vectors $\mathbf{d}_1(s), \mathbf{d}_2(s)$ are located in its normal plane. In the rest of the paper we assume that all variables and vectors are functions of the arc length s without explicitly restating this assumption.

At any point on the rods we introduce a twist vector $\kappa = (\kappa_1, \kappa_2, \kappa_3)$. $\kappa_i, i = 1, 2, 3$, is the component along the ith local basis vector and κ_1 and κ_2 represent the rotation angle per unit arc length along the two local basis vectors in the cross-section of rods, respectively, while κ_3 represents the twist angle per unit arc length along the tangent of rods. The twist vector can also be expressed as $\kappa = \kappa_1\mathbf{d}_1 + \kappa_2\mathbf{d}_2 + \kappa_3\mathbf{d}_3$ in the general fixed frame.

The generalized Frenet equations can be written as follows, cf. [7]:

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathbf{d_i} = \kappa \times \mathbf{d_i}, i = 1, 2, 3 \tag{1}$$

Using the matrix form of the equations and by introducing Euler angles (α, β, γ) for local basis vectors, $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$ we obtain

$$\begin{cases} \kappa_1 = -\frac{\partial\alpha}{\partial s}\cos\gamma\sin\beta + \frac{\partial\beta}{\partial s}\sin\gamma \\ \kappa_2 = \frac{\partial\alpha}{\partial s}\sin\gamma\sin\beta + \frac{\partial\beta}{\partial s}\cos\gamma \\ \kappa_3 = \frac{\partial\alpha}{\partial s}\cos\beta + \frac{\partial\gamma}{\partial s} \end{cases} \tag{2}$$

Another geometric relation about rods is the following:

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathbf{R} = \mathbf{d} \tag{3}$$

When the tangent director of rods is determined, eqn. 3 can be integrated to get the position of rods.

2.2 Introduction to the Static Kirchhoff Equations and its Boundary Value Problems

Consider an infinitesimal element of rods, using Newton's second law, we can obtain the equilibrium equations of rods [7]. In our work we will focus on the static case with all distributed loads ignored. Then the control equations of rods in the case of our interest are

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathbf{F} = \mathbf{0} \tag{4}$$

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathbf{M} + \mathbf{d_3} \times \mathbf{F} = \mathbf{0} \tag{5}$$

where **F** and **M** are the tension and internal moment of rods, respectively.

In this paper we consider linear material laws and the constitutive relationship of linear elasticity is given in terms of **M** and κ. This can be stated as follows:

$$\mathbf{M} = EI_{\mathbf{1}}\kappa_{\mathbf{1}}\mathbf{d_1} + EI_{\mathbf{2}}\kappa_{\mathbf{2}}\mathbf{d_2} + \mu J\kappa_{\mathbf{3}}\mathbf{d_3} \tag{6}$$

where E is elastic module, μ is shear module, I_1 and I_2 are the principal moment of inertia of the cross-section of rods, J is a function of shape of the cross-section of rods. In particular, for circular cross-sections, we have

$$I_1 = I_2 = \frac{J}{2} = \frac{\pi R^4}{2}$$

The rods can undergo large displacement, even when a linear constitutive relationship is used, cf. [7].

Following [7] eqns. 4, 5, and 6 can be rewritten in a scaled form, if the assumption of circular cross-section of rods is used:

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathbf{F} = \mathbf{0} \tag{7}$$

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathbf{M} + \mathbf{d_3} \times \mathbf{F} = \mathbf{0} \tag{8}$$

$$\mathbf{M} = \kappa_1\mathbf{d_1} + \kappa_2\mathbf{d_2} + b\kappa_3\mathbf{d_3} \tag{9}$$

where $b = \frac{\mu J}{EI_1} = \frac{1}{1+\nu}$and υ is Poisson's ratio.

The equations 1, 3, 7, 8, and 9 constitute a closed system consisting of seven vectors of dependent variables, as **F**, **M**, κ, $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$, **R**. However not all of the seven vectors are needed when solving the system. If we choose $\mathbf{F}, \kappa, \mathbf{d}_1, \mathbf{d}_2$ as dependent variables and write the corresponding equations extracted from the whole system in a component form, then we obtain

$$\begin{aligned}
\frac{\mathrm{d}F_1}{\mathrm{d}s} + \kappa_2 F_3 - \kappa_3 F_2 &= 0\\
\frac{\mathrm{d}F_2}{\mathrm{d}s} + \kappa_3 F_1 - \kappa_1 F_3 &= 0\\
\frac{\mathrm{d}F_3}{\mathrm{d}s} + \kappa_1 F_2 - \kappa_2 F_1 &= 0
\end{aligned}$$

$$\begin{aligned}\frac{\mathrm{d}\kappa_1}{\mathrm{d}s} + \kappa_2\kappa_3(b-1) - F_2 &= 0\\ \frac{\mathrm{d}\kappa_2}{\mathrm{d}s} + \kappa_1\kappa_3(1-b) + F_1 &= 0\\ \frac{\mathrm{d}}{\mathrm{d}s}\mathbf{d_1} = \boldsymbol{\kappa_3}\mathbf{d_2} - \boldsymbol{\kappa_2}\mathbf{d_1} &\times \mathbf{d_2}\\ \frac{\mathrm{d}}{\mathrm{d}s}\mathbf{d_2} = \boldsymbol{\kappa_1}\mathbf{d_1} \times \mathbf{d_2} &- \kappa_3\,\mathbf{d_1}\end{aligned}$$

where the tension of rods is expressed as $\mathbf{F} = F_1\mathbf{d}_1 + F_2\mathbf{d}_2 + F_3\mathbf{d}_3$, F_i, i =1,2 3, is the ith component of the tension of rods given in local frames. The reduced system can also be written in a concise form as follows:

$$\frac{\mathrm{d}\mathbf{Y}}{\mathrm{d}s} = \Xi(F_1, F_2, F_3, \kappa_1, \kappa_2, \kappa_3, \mathbf{d_1}, \mathbf{d_2})$$

where $\mathbf{Y} = (F_1, F_2, F_3, \kappa_1, \kappa_2, \kappa_3, \mathbf{d}_1, \mathbf{d}_2)$. With given boundary conditions we can define a boundary value problem associated with the static Kirchhoff rods. For example, at one end of rods, we can have the director vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ and at other end of rods we can give stresses. However, because we do not know the orientation at the right end of rods a priori, in general we cannot give the curvatures and forces there in component form directly. But we can still give the linear and angular momentum balances at that point, which will couple all the dependent variables. In the paper we will first deal with this case of boundary conditions.

In addition, the tangent director can be obtained by $\mathbf{d}_3 = \mathbf{d}_1 \times \mathbf{d}_2$; and $\mathbf{M}$ can be determined by using eqn. 9; the position vector can be obtained by integrating eqn. 3.

2.3 Closed Form of Solution of Euler Angles

Using 2, in which curvatures and local triad of rods are represented in term of Euler angles, eqn. 7, 8, and 9 are converted into the following three equivalent equations [7]:

$$\frac{\mathrm{d}\alpha}{\mathrm{d}s} = \frac{M_z - M_3 z}{1 - z^2} \tag{10}$$

$$\frac{\mathrm{d}\gamma}{\mathrm{d}s} = \left(\frac{1}{b} - 1\right)M_3 + \frac{M_3 - M_z z}{1 - z^2} \tag{11}$$

$$\left(\frac{\mathrm{d}z}{\mathrm{d}s}\right)^2 = 2F(h-z)(1-z^2) - (M_z - M_3 z)^2 \tag{12}$$

where $z = \cos\beta$, $h = \frac{1}{F}\left(H - \frac{M_3^2}{2b}\right)$ and F, M_z, M_3, H are constant system integrals, which are defined as

$$\begin{aligned} M_z &= \mathbf{M} \cdot \mathbf{e_z}\\ M_3 &= \mathbf{M} \cdot \mathbf{d_3}\\ H &= \frac{1}{2}\mathbf{M} \cdot \kappa + \mathbf{F} \cdot \mathbf{d_3}\end{aligned}$$

Here M_z represents the component of moments projected along axis z; M_3 represents the component of moments projected along tangent director $\mathbf{d}_3$; H represents the elastic energy function of rods. And F is the magnitude of the constant tension of rods which can be inferred from eqn. 7. In the following analysis, we assume that without loss of generality $\mathbf{F}$ is along axis $\mathbf{Z}$ of the fixed general frame.

The above three equations are not fully coupled. One can easily see that α and γ can be obtained by directly integrating eqn. 10 and 11 as seen beneath if the function z is known.

$$\alpha = \int_0^s \frac{M_z - M_3 z(\sigma)}{1 - z(\sigma)^2} \mathrm{d}\sigma + \alpha_0 \tag{13}$$

$$\gamma = \int_0^s \left[\left(\frac{1}{b} - 1 \right) M_3 + \frac{M_3 - M_z z(\sigma)}{1 - z(\sigma)^2} \right] \mathrm{d}\sigma + \gamma_0 \tag{14}$$

where α_0 and γ_0 are the integration constants of Euler angles α and γ. We refer to the paper [7] for more details on the explicit form of solutions of eqn. 12. However in [7], the author only expressed the solution for a special boundary conditions. This special case is not convenient to be used when specific initial values are given. In our work we will give a closed of form solution that is easy to use. It is assumed that the initial condition of eqn. 12 is, at s=0, $z = z_0 = \cos \beta_0$, where β_0 is the initial value of Euler angle β given at a boundary point. Then the explicit form of solution to eqn. 12 can be given as

$$z(s) = z_1 + (z_2 - z_1)\mathrm{JacobiSN}(\lambda(s + s_0), k)^2 \tag{15}$$

where $\lambda = \sqrt{\frac{F(z_3 - z_1)}{2}}$, $k = \sqrt{\frac{z_2 - z_1}{z_3 - z_1}}$ and s_0 can determined by the given initial conditions. And it is the root of the following equation:

$$\mathrm{JacobiSN}(\lambda s_0, k) = \sqrt{\frac{z_0 - z_1}{z_2 - z_1}} \tag{16}$$

z_1, z_2 and z_3 are the three real roots of the cubic polynomial of the right hand side of eqn. 12 at z. These roots are assumed to be in the following order [7].

$$-1 \le z_1 \le z_2 \le 1 \le z_3 \tag{17}$$

JacobiSN(x, k) is one of the Jacobi's elliptic functions [7]. One can easily prove in a symbolic system such as Maple that eqn. 15 is really the solution to eqn. 12. According to the above analysis one can see that if at the boundary of rods the three Euler angles α_0, β_0, γ_0, and the material constant b and the four system constants F, M_z, M_3 and H are given, then all dependent variables of the system can be determined.

3 Boundary Matching Method

According to the analysis in Sec. 2, the static Kirchhoff equations can be fully determined if all the necessary parameters are known. However, in general these parameters do not obviously appear in the equations of boundary conditions. In the following we develop a method for locating all these parameters which make the explicit form of solution match the given boundary conditions. In this way the BVPs associated with the static Kirchhoff rods are solved.

In particular, in eqns. 13–15, α_0, γ_0 and z_0 are related to the orientation of the boundary points of rods. And the parameter b can be calculated by using the given physical parameters. Normally the four constant system integrals, F, M_z, M_3 and H cannot all be determined directly. In the following, we present a unified boundary matching method to deal with two commonly used boundary conditions.

3.1 First Case of Boundary Conditions

For the first case of interest, as is shown in Fig. 1, at the left end of rods, say $s = 0$, the position vector $\mathbf{R}_L$ is given and also the local basis vectors $\mathbf{d}_{1L}$, $\mathbf{d}_{2L}$, $\mathbf{d}_{3L}$. At the right end of rods, say $s = L$ (length of rods), external forces $\mathbf{F}_R$ and torques $\mathbf{M}_R$ are exerted.

All the given boundary conditions are expressed in the general fixed frame now. However, we still need another fixed reference frame $\mathbf{XYZ}$, the axis of which is related to the direction of the tension of rods and also in which the closed form of the solution of our system is given as seen in Fig. 1. All the given boundary conditions will be represented in this reference frame after it is built. Let us choose the axis $\mathbf{Z}$ of the reference frame oriented along the external force exerted at the right end of rods and take the left end of rods as its origin. It is easy to see that the angle between axis $\mathbf{Z}$ and the tangent director $\mathbf{d}_3$ at the starting point of rods is the initial value of Euler angle β, the cosine of which is equal to z_0. The axis $\mathbf{Y}$ will be the cross-product of vector $\mathbf{Z}$ and $\mathbf{d}_{3L}$. Then one can easily get the axis $\mathbf{X}$ of the reference frame $\mathbf{X} = \mathbf{Y} \times \mathbf{Z}$. Next we can use the local triad given at the left end of rods to determine the initial values of the other two Euler angles, α and γ. To do so we set another local reference frame at the point, say $\mathbf{d}'_{1L}\mathbf{d}'_{2L}\mathbf{d}'_{3L}$. The axis $\mathbf{d}'_{3L}$ of the local reference frame coincides with the axis $\mathbf{d}_{3L}$ and the axis $\mathbf{d}'_{2L}$ coincides with the axis $\mathbf{Y}$ of the reference frame $\mathbf{XYZ}$. Then we have $\mathbf{d}'_{1L} = \mathbf{d}'_{2L} \times \mathbf{d}'_{3L}$. Accordingly, the three Euler angles of the local reference triad $\mathbf{d}'_1\mathbf{d}'_2\mathbf{d}'_3$given in the reference frame $\mathbf{XYZ}$ are$\alpha_0 = \gamma_0 = 0, \beta_0 = \arccos(z_0)$, respectively. One can see that the only difference between the given local frame $\mathbf{d}_{1L}\mathbf{d}_{2L}\mathbf{d}_{3L}$ and local reference frame $\mathbf{d}'_{1L}\mathbf{d}'_{2L}\mathbf{d}'_{3L}$ is a rigid rotation along axis $\mathbf{d}_{3L}$. Thus the initial values of the three Euler angles of the local triad $\mathbf{d}_{1L}\mathbf{d}_{2L}\mathbf{d}_{3L}$ are $\alpha_0 = 0, \beta_0 = \arccos(z_0), \gamma_0 = \overline{\gamma}_0$, respectively, where $\overline{\gamma_0}$ is the angle by which the axis $\mathbf{d}'_{1L}$ is rotated to axis $\mathbf{d}_{1L}$ along axis $\mathbf{d}_{3L}$.

Now all boundary conditions given at the starting point have been used. And at the other end of rods, the external force condition $\mathbf{F}_R$ has been used too, because its direction is used for setting the axis $\mathbf{Z}$ and its magnitude is equal to one of the four constant system integrals, namely F. In addition at any point of rods, say s, the internal moment $\mathbf{M}_s$ can be expressed as

$$\mathbf{M}_s = (\mathbf{\Delta R}_s) \times \mathbf{F}_R + \mathbf{M}_R$$

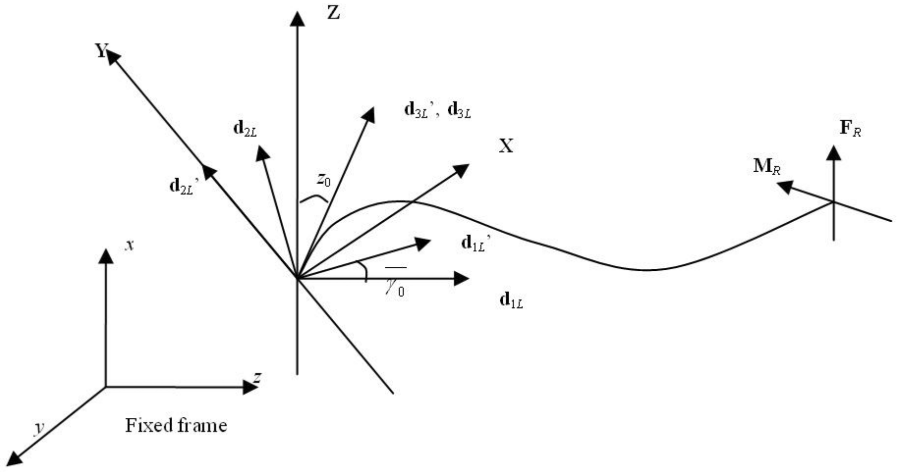


Fig. 1. Boundary matching scheme

where $\varDelta\mathbf{R}_s = \mathbf{R}_R - \mathbf{R}_s$, $\mathbf{R}_R$ and $\mathbf{R}_s$ are the position vector of the right end of rods and the point s, respectively. Projecting both sides of the above equation along axis $\mathbf{Z}$, one can get the constant system integral M_z,

$$\mathbf{M}_z = (\mathbf{M}_R) \cdot \mathbf{e}_z$$

However M_3 can not be obtained in this way, because $\varDelta\mathbf{R}_s$ can not be determined. Neither the constant system integral H can be determined using the given boundary conditions. But if these two system constants are determined all the descriptive variables of Kirchhoff rods can be determined. Thus we may take them as unknown parameters to be determined. In the following we give a method for finding appropriate values for the two parameters to match the given boundary conditions.

So far the only boundary condition that has not been used is the moment exerted at the right end of rods, which can be equivalently written as

$$\begin{cases} \kappa_1(L) - \overline{\kappa}_1(L) = 0 \\ \kappa_2(L) - \overline{\kappa}_2(L) = 0 \\ \kappa_3(L) - \overline{\kappa}_3(L) = 0 \end{cases} \tag{18}$$

where $\kappa_i(L)$ and $\overline{\kappa}_i(L)$, i=1, 2, 3, represent the curvature along the i^{th} axis at point $s = L$. The former is determined by using eqn. 2, while the latter is calculated by using the constitutive relationship and the exerted moment there, obtained by the following formulae.

$$\begin{cases} \overline{\kappa}_1(L) = \ \ \mathbf{M}_R \cdot \mathbf{d_1(L)} \\ \overline{\kappa}_2(L) = \ \ \mathbf{M}_R \cdot \mathbf{d_2(L)} \\ \overline{\kappa}_3(L) = \mathbf{M}_R \cdot \mathbf{d_3(L)/b} \end{cases}$$

constructing a function as follows:

$$f_1 = \frac{1}{2}\left\{[\kappa_1(L) - \overline{\kappa}_1(L)]^2 + [\kappa_2(L) - \overline{\kappa}_2(L)]^2 + [\kappa_3(L) - \overline{\kappa}_3(L)]^2\right\} \tag{19}$$

One can easily see that it is in fact a function of the constant system integrals M_3 and H. In addition—from the definition of the function—one can see that the conditions minimizing the function 19 are necessary and sufficient for eqn. 18 to be fulfilled, and vice versa. Thus the problem of matching the last boundary conditions, eqn. 18, is converted to finding appropriate M_3 and H that minimize the function f_1 from 19. Because we are going to deal with an unconstraint minimization problem, the two unknown parameters M_3 and H could be in the range of the whole real region. However H can not take all real numbers since it can be expressed in term of Euler angles as in [7],

$$H = \frac{1}{2}\left[(\theta')^2 + \frac{M_3^2}{b} + \frac{(M_z - M_3 z)^2}{1 - z^2}\right] + F_z \tag{20}$$

where $\theta' = \left(\frac{\mathrm{d}\theta}{\mathrm{d}s}\right)$. Eqn. 21 indicates that H can only be in some regions of the reals. However by observation of eqn. 21, we found that θ' can be any real number. Thus we select θ' as the other unknown parameter for our minimization scheme in place of H. Since H is a constant system integral which is of course not dependent on arc length, we can extract it at the starting point where the Euler angle β_{w} and $P = \theta'(0)$ and other necessary parameters are known in this case. Then M_3 and $P$are selected as the unknown parameters for our minimization problem. A global minimization approach can be well chosen to find appropriate values of M_3 and P with which the explicit form solution of our system can match the given boundary conditions. We refer to the first example in Sec. 4.

3.2 The Second Case of Boundary Conditions

In this section we use similar technique to deal with another case of boundary conditions. At the left end of rods, say $s = 0$, the position vector $\mathbf{R}_L$ is given and also the local triad of the point $\mathbf{d}_{1L}$, $\mathbf{d}_{2L}$, $\mathbf{d}_{3L}$. At the other end of rods only the position vector $\mathbf{R}_R = (x_R, y_R, z_R)$ is given, written as,

$$\begin{cases} x(L) = x_R \\ y(L) = y_R \\ z(L) = z_R \end{cases} \tag{21}$$

where $\big(x(L)\ y(L)\ z(L)\big)$ is the position vector at the right end of rods which is the result of integrating eqn. 3. Similarly we first construct a cost function f_2,

$$f_2 = \frac{1}{2}\left\{[x(L) - x_R]^2 + [y(L) - y_R]^2 + [z(L) - z_R]^2\right\} \tag{22}$$

According to the definition of the function 22 the conditions that minimize the function 22 are equivalent to the boundary condition in eqn. 21.

For this case of boundary conditions we choose M_z, M_3, β_0 and P ($P = \theta'(0)$) as the unknown parameters to be determined. This case is a little bit more complex than the previous one because all these parameters cannot be determined directly by using the given boundary conditions. The constant system integral F is in this case taken as an external parameter. Then we can examine how the closed form of solution matches the given boundary conditions at different level of tension.

When the cost function is evaluated M_z, M_3, β_0 and P will be passed to it as input arguments. Similarly we first build a reference frame $\mathbf{XYZ}$. Without loss of generality, we choose the left end of rods as its origin. Then we set the axis $\mathbf{Y}$ of the reference frame oriented along the direction that is perpendicular to the tangent director $\mathbf{d}_{3L}$ at the left end of rods just like that of Sec. 3.1. The axis $\mathbf{Z}$ is set as a vector which will coincide with the tangent director $\mathbf{d}_{3L}$ at the left end of rods if it is rotated by angle β_0 along axis $\mathbf{Y}$. Then one can easily get $\mathbf{X} = \mathbf{Y} \times \mathbf{Z}$. After the reference frame is obtained, one can similarly determine the other two initial values of Euler angles α_0 and γ_0 using the local triad given at the left end of rods. Thus the problem of matching the boundary condition, eqn. 21 is converted to solving an unconstrained minimization problem with M_z, M_3, β_0 and P as unknown parameters to be determined by minimizing f_2.

4 Example

Our work is motivated by human hair modeling. In the following example we assume that the rods considered will have the physical properties of human hair: the Young's module of rods is 3.89e10 dyne/cm^2; the Poisson's ratio is 0.25; the radius of rods is 0.005 cm [15].

For the example, we consider the first case of boundary conditions in which the left end of a rod is fixed at the origin and the director vectors are also given, $\mathbf{d}_1 = (0.0, 0.0, 1.0)$, $\mathbf{d}_2 = (1.0, 0.0, 0.0)$; and at the right end of the rod, external forces and torques are exerted, $\mathbf{F}_R = (0.0, 0.0, -2.0)$, $\mathbf{M}_R = (0.0, 0.0, 5.0)$. We will consider several rods with various length in the example, say $L=$ 10cm, 20cm, 30cm, respectively.

We use the multidimensional downhill simplex method for our minimization task [16]. Although the algorithm is not a global minimization method, from the definition of our cost function, one can easily know that it is non-negative in the whole real region. It can be inferred that the global minimum of our cost function is zero. Thus we can easily check if the results of the minimization scheme are the desired. In addition, we do not need to evaluate the derivative of the cost function in the minima finder, which can only be calculated numerically in our method. On the other side, we also use a global minimization method for the purpose, called Sigma [17]. However in our experience, it will be very time consuming if we use the global method. The cost function will be evaluated for about several tens of thousands or even hundreds of thousands times. However similar results compared with those of the downhill simplex method were obtained, in which only several hundreds of times of evaluation of the cost function are needed. In Table 1, we give the results of our computations of several rods with different length under the case of boundary conditions.

In Fig. 2 (a) to (c), we show the final shape of the rods for three cases.

Table 1. Results of our minimization method for the example

The CPU times were measured on a Pentium IV 2.2 GHz PC.

Length (cm)		10	20	30
Values at start	(M3, P)	(-2.0, -2.0)	(-0.260, 0.375)	(-0.262, 0.375)
	Function value	5.091	0.03337	0.2223
Values at end	(M3, P)	(-0.2583, -0.3740)	(-0.26183, 0.37538)	(-0.2618, 0.3753)
	Function value	0.0	0.0	0.0
Number of function evaluations		219	179	220
Total CPU time (approximate)		0.5 sec	0.5 sec	0.5 sec

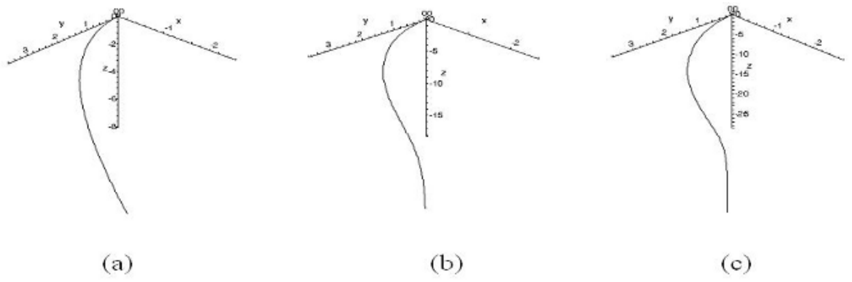


Fig. 2. The shapes of a rod with a variety of lengths; (a) length: 10 cm; (b) length: 20 cm; (c) length: 30 cm

5 Conclusion

In this paper we presented a symbolic-numeric method for solving various boundary value problems associated with the static Kirchhoff rods. We first expressed the explicit form of solution to the static Kirchhoff rod equations in a form which is easily parameterized by initial values. By combining the parameterized closed form solution with a global minimization scheme we presented a general method in which the problem of solving the boundary value problems associated with static Kirchhoff rods is converted to the problem of solving an unconstrained global minimization problem. Our method can be used to adapt a variety of boundary conditions, as we only need to construct different cost functions for them. An adaptation to other constraints and boundary conditions, e.g. the ones arising from contact between hair fibers, will be a topic of our future research.

Acknowledgements. The work was financially supported by *Deutsche Forschungsgemeinschaft* under grant We 1945/3-1.

References

1. Chin, C.H.K., May, R.L., Connell, H.J.: A numerical model of a towed cable-body system. J. Aust. Math. Soc. **42(B)** (2000) 362–384
2. Swigon, D., Coleman, B.D., Tobias, I.: The elastic rod model for DNA and its application to the tertiary structure of DNA minicircles in mononucleosomes. Biop. J. **74** (1998) 2515–2530
3. Colemana, B.D., Olsonb, W.K., Swigonc, D.: Theory of sequence-dependent DNA elasticity. J. Chem. Phys. **118** (2003) 7127–7140
4. Moakher, M., Maddocks, J.H.: A double-strand elastic rod theory. In: Workshop on Atomistic to Continuum Models for Long Molecules and Thin Films. (2001)
5. Coleman, B.D., Swigon, D.: Theory of self-contact in Kirchhoff rods with applications to supercoiling of knotted and unknotted DNA plasmids. Philosophical Transactions: Mathematical, Physical and Engineering Sciences **362** (2004) 1281–1299
6. Goriely, A., Tabor, M.: Spontaneous helix-hand reversal and tendril perversion in climbing plants. Phys. Rev. Lett. **80** (1998) 1564–1567
7. Nizette, M., Goriely, A.: Towards a classification of Euler-Kirchhoff filaments. Journal of Mathematical Physics **40** (1999) 2830–2866
8. Goriely, A., Nizette, M., Tabor, M.: On the dynamics of elastic strips. Journal of Nonlinear Science **11** (2001) 3–45
9. da Fonseca, A.F., de Aguiar, M.A.M.: Solving the boundary value problem for finite Kirchhoff rods. Physica D **181** (2003) 53–69
10. Thomas, Y.H., Klapper, I., Helen, S.: Romoving the stiffness of curvature in computing 3-d filaments. J. Comp. Phys. **143** (1998) 628–664
11. Rappaport, K.D.: S. Kovalevsky: A mathematical lesson. American Mathematical Monthly **88** (1981) 564–573
12. Goriely, A., Nizettey, M.: Kovalevskaya rods and Kovalevskaya waves. Regu. Chao. Dyna. **5(1)** (2000) 95–106
13. Shi, Y., Hearst, J.E.: The Kirchhoff elastic rod, the nonlinear Schroedinger equation and DNA supercoiling. J. Chem. Phys. **101** (1994) 5186–5200
14. Pai, D.K.: STRANDS: Interactive simulation of thin solids using Cosserat models. Computer Graphics Forum **21** (2002) 347–352 EUROGRAPHICS 2002.
15. Robbins, C.R.: Chemical and physical behavior of human hair. Springer (2002)
16. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, Second Edition. Cambridge University Press (2002)
17. Aluffi-Pentini, F., Parisi, V., Zirilli, F.: Sigma — a stochastic-integration global minimization algorithm. ACM Tran. Math. Soft. **14** (1988) 366–380

Differential and Difference Equations for Products of Classical Orthogonal Polynomials

Sergey Slavyanov and Vladimir Papshev

St-Petersburg State University, V.A. Fock Institute of Physics,
St-Petersburg 198506, Botanicheskaya 1, Russia

Factorization of differential equations has been intensively studied (see, for instance, [1], [2]). Less results are known for difference equations. In this publication we are not giving a general approach to the theory of factorization but rather present some observations and derive formulae for further use in reference books and for symbolic computations.

Several specific examples which arise from the theory of classsical orthogonal polynomials are studied. They have, to our mind, significance for practical applications in physics.

The paper is based to some extent on the ideas developed in other publications of the authors [3], [5], [6] but the angle of view on the problem is different. In the first section differential equations are dealt with. In the second section, our studies are concentrated on difference equations. In both cases knowing the equation for orthogonal polynomials we derive equations for their products. These latter equations are of higher order than the starting ones, and polynomial solutions can be sought as solutions of multiparametric spectral problem [4].

1 Differential Equations for the Products of Orthogonal Polynomials

Classical orthogonal polynomials may be considered as appropriate solutions (related to eigenfunctions of a corresponding singular Sturm–Liouville problem) of the particular cases of the linear second-order differential equation

$$r(x)y''(x) + p(x)y'(x) + \lambda y(x) = 0. \tag{1}$$

If

$$\begin{aligned} r(x) = 1 - x^2, \quad p(x) = (b+1)(1-x) - (a+1)(1+x),& \\ \lambda_n = n(n+a+b+1)& \end{aligned} \tag{2}$$

(1) generates Jacobi polynomials. If

$$r(x) = 1 - x^2, \quad p(x) = -2x, \quad \lambda_n = n(n+1) \tag{3}$$

(1) generates Legendre polynomials. If

$$r(x) = x, \quad p(x) = a + 1 - x, \quad \lambda_n = n \tag{4}$$

(1) generates Laguerre polynomials. If

$$r(x) = 1, \quad p(x) = -2x, \quad \lambda_n = n \tag{5}$$

(1) generates Hermite polynomials.

An auxiliary equation to (1) can be written as

$$r(x)u''(x) + p(x)u'(x) + \mu u(x) = 0 \tag{6}$$

with other value μ of the spectral parameter. Multiplying (1) by $u(x)$ and (6) by $y(x)$ we obtain the following equation

$$r(x)v''(x) + p(x)v'(x) + \sigma v(x) = 2r(x)y'u', \quad \text{with } \sigma = \lambda + \mu \tag{7}$$

The left-hand side of (7) contains only the product $v = yu$ of solutions of (1) and (6) whenever the right-hand side of (7) still contains these solutions. The further trick is "to kill" the latter terms (they are called below "unbalanced terms") by subsequent differentiation of (7).

To do so is easy in the self-adjoint case of (1,7) when $p(x) = r'(x)$. After two sequential differentiations of (7) the unbalanced terms can be excluded, and the following equation holds as the result

$$(r(rv')')'' + \sigma[(r'v)' + 2(rv')'] + \Delta^2 v = 0, \tag{8}$$

where

$$\Delta_l = \lambda_n - \lambda_m, \; l = |n - m|,$$
$$\sigma_j = \lambda_n + \lambda_m, \; j = n + m.$$

In the case of Legendre polynomials $p(x) = r'(x)$ and the corresponding equation (1,3) is in the self-adjoint form. Hence, (8) solves the problem with products of Legendre polynomials.

If (1) is not in the self-adjoint form like for (2,4,5), it is needed to introduce an "integrating multiplier" before the second differentiation.

Lemma 1. *It is possible to eliminate the unbalanced terms introducing at the second differentiation the "integrating multiplier" ω. The integrating multiplier ω is found from the first-order equation*

$$\frac{\omega'}{\omega} = \frac{p - r'}{r}. \tag{9}$$

Proof. The unbalanced terms arising after first differentiation of (7) are

$$2r(\lambda y u' + \mu y' u).$$

Consider the expression

$$\tau = -2\omega r(\lambda y u' + \mu y' u).$$

Condition (9) leads to the consequence that the function τ' contains no more unbalanced terms and is expressed in terms of v

$$\tau' = \omega(-\sigma(rv'' + pv') - \Delta^2 v).$$

Two remarks are needed: 1) the integrating multiplier is equal to the weight function for considered polynomials, 2) the integrating multiplier up to a constant factor is unique.

Boring computations with Maple lead to the following equation for the product

$$r^3 v'''' + r^2(4p + r')v''' + r\left[(2r(2p' + \sigma) + p(5p - r')\right] v'' + \\ \left[r^2 p'' + r(5p'p - (pr')' + \sigma(4p - r') + p(p - r')(2p - r'))\right] v' \\ + \left[r((2p' - r'')\sigma + \Delta^2) + (p - r')(2p - r')\sigma\right] v = 0. \tag{10}$$

Equation (10) solves the above formulated problem for other types of classical orthogonal polynomials.

The simplest case are Hermite polynomials. For them the following fourth-order differential equation holds

$$v'''' - 8xv''' + [(2(\sigma - 4) + 20x^2]v'' + \\ \left[20x - 8\sigma x - 16x^3\right] v' + \left[-4\sigma + \Delta^2 + 8x^2\sigma\right] v = 0. \tag{11}$$

The obtained differential equations can be helpful for the following purposes.

1) They give representative examples for factorization of higher order differential equations.
2) They give examples of fourth-order equations with polynomial solutions.
3) If some integral transform is applied to these equations several other equations with more or less simple explicit solutions can be generated. It could be, for instance, multipole matrix elements in quantum physics [6].
4) They give examples of solvable multiparametric eigenvalue problems, which regretfully are not self-adjoint.

2 Difference Equations for Products of Classical Orthogonal Polynomials

Each classical orthogonal polynomial $y_n(x)$ can be studied as a function of continuous variable x and index n. On the other hand it can be considered as a function $y(n, x)$ of the discrete variable n on an integer grid and of the parameter x. Below the notation $y(n)$ is used instead of $y(n, x)$. The shift operator E can be defined on $y(n)$

$$Ey(n) = y(n + 1).$$

In these notations polynomials $y(n)$ satisfy the following difference equation [7]

$$\alpha(n)Ey(n) = (kx + \beta(n))y(n) + \gamma(n)E^{-1}y(n) \tag{12}$$

with polynomial coefficients in n

$$\alpha = n + 1, k = -1, \quad \beta = (2n + a + 1), \quad \gamma = -(n + a) \tag{13}$$

in the case of Laguerre polynomials,

$$\alpha = n+1,\ k = 2n+1, \beta = 0, \quad \gamma = -n \tag{14}$$

in the case of Legendre polynomials,

$$\alpha = 1,\ k = 2,\ \beta = 0,\ \gamma = -2n \tag{15}$$

in the case of Hermite polynomials and more complicated expressions

$$\begin{aligned} \alpha &= 2(n+1)(n+a+b+1)(2n+a+b), \\ k &= (2n+a+b+2)(2n+a+b+1)(2n+a+b), \\ \beta &= (2n+a+b+1)(a^2-b^2), \\ \gamma &= -2(n+a)(n+b)(2n+a+b+2) \end{aligned} \tag{16}$$

in the case of Jacobi polynomials. It is needed to point out that (13)–(16) depend on normalization of chosen polynomials

$$\int_a^b w(x) f_n^2(x) dx = h_n,$$

where $w(x)$ is the weight function, and f_n are the studied polynomials. The value h_n equals

$$h_n = \frac{\Gamma(n+a+1)}{n!}, \quad a > -1$$

in the case of Laguerre polynomials,

$$h_n = \frac{2}{2n+1}$$

in the case of Legendre polynomials,

$$h_n = \sqrt{\pi} 2^n n!$$

in the case of Hermite polynomials, and

$$h_n = \frac{2^{a+b+1}}{2n+a+b+1} \frac{\Gamma(n+a+1)\Gamma(n+b+1)}{\Gamma(n+1)\Gamma(n+a+b+1)}$$

in the case of Jacobi polynomials.

The auxiliary difference equation reads

$$\alpha(m)Eu(m) = (kx + \beta(m))u(m) + \gamma(m)E^{-1}y(m). \tag{17}$$

The product of the solutions of (12) and (17) is defined as

$$v(n,m) = y(n)u(m). \tag{18}$$

It is assumed below that the shift operator acts on both arguments of $v(n,m)$ simultaneously, so, for instance, $Ev(n,m) = v(n+1,m+1)$ and $E^{-1}v(n,m) = v(n-1,m-1)$. Multiplying (12) and (17) we obtain

$$\begin{aligned}\alpha(n)\alpha(m)Ev - (kx+\beta(n))(kx+\beta(m))v - \\ \gamma(n)\gamma(m)E^{-1}v = -(kx+\beta(n))\gamma(m)y(n)E^{-1}u - \\ (kx+\beta(m))\gamma(n)uE^{-1}y.\end{aligned} \tag{19}$$

Similar to (7) and to the equation which follows from (7) after differentiation we have "unbalanced terms" in the right-hand side of (19) whenever the expression in the left-hand side of (19) depends only upon the product $v(n,m)$. In order to wipe them out it is needed to apply operators E and E^{-1} to (19). After further manipulations with obtained formulae and initial equations (12,17) the fourth-order difference equation for $v(n,m)$ appears.

In the general case the computations are rather cumbersome and need use of Maple or other CAS. Hence, as a simple example the case of Hermite polynomials is studied. For them equation (19) reads

$$Mv : Ev - 4x^2v - 4nmE^{-1}v = -4x(my(n)E^{-1}u(m) + nu(m)E^{-1}y(n)). \tag{20}$$

Using the shifts E and E^{-1} two other equations can be derived from (20)

$$\begin{aligned}E^2v - 4x^2Ev - 4(n+1)(m+1)v = \\ -4x((m+1)Ey(n)u(m) + (n+1)y(n)Eu(m))\end{aligned} \tag{21}$$

and

$$\begin{aligned}v - 4x^2E^{-1}v - 4(n-1)(m-1)E^{-2}v = \\ -4x((m-1)E^{-1}y(n)E^{-2}u(m) + (n-1)E^{-2}y(n)E^{-1}u(m)).\end{aligned} \tag{22}$$

In (21) the "forward" substitution of (12) and (17) is performed with the result

$$\begin{aligned}M^+v : E^2v - 4x^2Ev + 4(2x^2(m+n+2) - (n+1)(m+1))v \\ = 8x((m+1)nu(m)E^{-1}y(n) + (n+1)my(n)E^{-1}u(m)).\end{aligned} \tag{23}$$

The purpose of this substitution was to eliminate terms $Ey(n)$ and $Eu(m)$ from the right-hand side expression in (21). The "backward" substitution of (12) and (17) into (22) leads to the difference equation

$$\begin{aligned}M^-v : v + 4x^2E^{-1}v - 4(n-1)(m-1)E^{-2}v = \\ 2x(u(m)E^{-1}y(n) + y(n)E^{-1}u(m)).\end{aligned} \tag{24}$$

After elimination of unbalanced terms $y(n)E^{-1}u(m)$ and $y(n)E^{-1}u(m)$ from (20, 23, 24) the following five-term difference equation is obtained

$$(M^+ - 4mnM^- + 2M)v = 0. \tag{25}$$

The final equation follows from explicit expressions for M, M^+, and M^-

$$E^2v - 2(2x^2-1)E^{-1}v + (4(2x^2-1)(m+n+1) - 8mn)v - \\ 8nm(2x^2+1)E^{-1}v + 16mn(n-1)(m-1)E^{-2}v = 0, \qquad (26)$$

which gives a simple example of factorization of difference equations.

It is known that recurrence (12) gives one of the most effective tools for symbolic calculation of classical orthogonal polynomials. We claim that recurrence (26) for Hermite polynomials (and similar recurrences for other polynomials) give an effective algorithm to calculate products of these polynomials which is better compared to multiplication. How can this algorithm be organized? Suppose it is needed to calculate $v(n,m) = H_N(x)H_M(x)$. Let $N \geq M$ and $k = N - M$. First with the help of (12,15) and multiplication the values of $v(k+1,1)$, $v(k+2,2)$, $v(k+3,3)$, $v(k+4,4)$ are found, and then within M steps we obtain the needed product.

We have performed the computations for the general case with Maple. However, the results are too bulky to be exposed in this text and will be presented elsewhere.

The field of application of the proposed scheme is broader than the classical orthogonal polynomials but in this broader field other aspects should be taken into account which are beyond the scope of this publication.

Technically it was possible to use in this section the difference operator Δ instead of the shift operator E. This would lead to closer relationship to the first section but would be unlike the notations used in the theory of orthogonal polynomials.

References

1. Berkovich, L.M.: Factorization and transformation of differential equations. R&C Dynamics (2002) (in Russian)
2. Schwartz, F.: A factorization algorithm for linear ordinary equations. In: Proc. ACM-SIGSAM (1989)
3. Slavyanov, S.Yu.: The equation for the product of solutions of two different Schrödinger equations. Theor. & Math. Phys. **136** (2003) 1251–1257
4. Volkmer, H.: Multiparameter eigenvalue problem and expansion theorems. Springer, New York, Heidelberg, Berlin (1988)
5. Slavyanov, S.Yu., Papshev, V.Yu.: Product of the Jacobi polynomials. Ulmer Seminare **8** (2003) 346–352
6. Slavyanov, S.Yu., Papshev V.Yu.: Explicit and asymptotic study of one-dimensional multipole matrix elements. In: “Nonadiabatic Transitions in Quantum Systems”, Chernogolovka (2004) 84–93
7. Abramowitz, M., Stegun, I.: Handbook of Mathematical Functions. NIST (1964)

Algorithm of Local Resolution of Singularities of a Space Curve*

Akhmadjon Soleev

Department of Mathematics, Samarkand State University,
Samarkand, 703004, Uzbekistan
asoleev@yandex.ru

Abstract. In this paper we present a procedure that allows us to distinguish all branches of a space curve near the singular point and to compute parametric form of them with any accuracy. The same procedure works for finding the branches of a space curve such that some (or all) coordinates tend to infinity.

Introduction. Let an algebraic curve F be defined in C^n by the system of polynomial equations

$$f_i(X) \overset{\text{def}}{=} \sum a_{iQ} X^Q = 0, \quad Q = (q_1, \ldots, q_n) \in D_i, \quad i = 1, \ldots, n-1, \tag{1}$$

where $D_i \overset{\text{def}}{=} D(f_i) = \{Q : a_{iQ} \neq 0\}$. Let $X = (x_1, \ldots, x_n) = 0$ be a singular point of F, i.e. all $f_i(0) = 0$ and $\operatorname{rank}(\partial f_i/\partial x_j) < n-1$ in $X = 0$. Then several branches of F pass through the $X = 0$. Each branch has its own local uniformization [5] of the form

$$x_i = \sum_{k=1}^{\infty} b_{ik} t^{p_{ik}}, \quad i = 1, \ldots, n \tag{2}$$

where exponents p_{ik} are integers, $0 > p_{ik} > p_{ik+1}$, and coefficients b_{ik} are complex numbers, series converge for large $|t|$, i.e. $X \to 0$ for $t \to \infty$. We propose an algorithm for finding any initial parts of the expansion (2) for all branches of F.

At first we find a list of truncated systems

$$\widehat{f}_i(X) \overset{\text{def}}{=} \sum a_{iQ} X^Q = 0, \quad Q \in D_{ij}^{(d_i)}(f_i), \quad i = 1, \ldots, n-1. \tag{3}$$

with the help of Newton polyhedra of polynomials f_i [4]. Each of them is the first approximation of (1). By the power transformation

$$y_i = x_1^{\alpha_{i1}} \ldots x_n^{\alpha_{in}}, \quad i = 1, \ldots, n \tag{4}$$

we reduce the number of variables in the truncated system(3). The power transformation (4) gives a method for the solution of truncated systems (3) and

* This work was supported by State Committee for Science and Technology of the Republic of Uzbekistan, grant No. 1.1.13.

resolves (only partly) the singularity $X = 0$ of system (1). In the transformed system (1), we find all points Y^0 corresponding to the point $X = 0$ of F. We translate each Y^0 into the origin and repeat the procedure described above. After a finite number of such steps, we come to a system having unique local branch, and we uniformize it by means of the Implicit Function Theorem. Returning to the initial coordinate X by inverse transformations we obtain the branch in the form (2). We uniformize similarly all other branches of the curve F near the origin $X = 0$ and all branches going to infinity and real branches of a real curve as well.

We could solve system (1) by eliminating $n-2$ coordinates, writing the system as a single equation $g(x_1, x_2) = 0$, and solving with Newton's method. But the process of eliminating variables leads to a great increase in the order of the polynomials involved [10]. Hence, the algorithm in this paper requires much less computation.

Graves [9] proposed (and Botashev [11] developed) considering the system (1) in the form of a single vector equation

$$G \overset{\text{def}}{=} \sum_{p,q,} G_q^{(p)} x^q = 0$$

with a single Newton polygon on the (p, q)-plane. A deficiency of this approach is that with Botashev's method certain components of the vector $\widehat{G}$ can be identically equal to zero in the truncated vector equation $\widehat{G} = 0$, while with each $\widehat{f}_i$ in the truncated system (3) contains some terms $g_{iq}^{(p)} x^q$ that are not identically zero. Therefore, with Botashev's method the sets of solutions of truncated vector equation can have larger dimension, and it makes more difficult to find even the branches with the homogeneous expansions (2). The calculations of Botashev method are, thus, simpler than those of the method of elimination, but are more complicated than the computation of our method.

Bernshtein [1] employed the approach to compute the number of branches of an algebraic curve of special form, but he did not consider the question of computing asymptotic expansions. In contrast to the papers of Bernshtein [1], Khovanskii [7], and Varchenko [6] it is not assumed in the present article that system (1) is in general position. To simplify the presentation it is assumed only that the system (1) does not belong to an exceptional set of infinite codimension in the space of systems (1). More about history of this problem see [2].

1°. Let us consider the finite sum of monomials

$$f(X) = \sum a_Q X^Q \tag{5}$$

without similar terms and $a_Q \in C$.

The set $D = \{Q : a_Q \neq 0\}$ is called the support of f. We assume that $D \subset \mathbf{Z}^n$, and we enumerate all points of D as $Q_1, \ldots, Q_l$. Let t be a real parameter. After the substitution

$$\begin{gathered} x_i = t^{p_i}, \quad p_i \in \mathbf{R}, \quad i = 1, \ldots, n, \quad P = (p_1, \ldots, p_n), \\ P \neq 0, \quad P \leq 0 \end{gathered} \tag{6}$$

into (5), we have

$$f(X) = \sum a_Q t^{\langle P,Q\rangle}, \tag{7}$$

where $\langle P, Q\rangle = p_1q_1 + \cdots + p_nq_n$ is the scalar product. Let us find main terms in sum (7) for $t \to \infty$, and let them for simplicity be the terms with the first k exponents $Q_i,\ 1 \le k \le l$, i.e.,

$$\begin{array}{l}\langle P, Q_1\rangle = \cdots = \langle P, Q_k\rangle, \\ \langle P, Q_1\rangle > \langle P, Q_j\rangle, \quad k < j < l+1.\end{array} \tag{8}$$

Let us denote $D_P = \{Q_1, \ldots, Q_k\}$, then the sum

$$\widehat{f}_i(X) \ = \ \sum a_Q X^Q, \text{where} \quad Q \in D_P \tag{9}$$

is called the truncation of f for the vector order P. After a substitution of the form

$$x_i = b_i t^{p_i}(1 + o(1)), \quad b_i \neq 0, \quad i = 1, \ldots, n, \tag{10}$$

where $t \to \infty$, we have $f(X) = t^r \widehat{f}(B) + o(t^r), \quad r = \langle P, Q_1\rangle$, $B = (b_1, \ldots, b_n)$.

The set $N = N(f, \widehat{f})$ of all those P for which $\widehat{f}$ is the truncation of f is the (normal) cone of the truncation (9). N is described by system (8).

In order to describe the sets D_p for different vectors P, we denote by M the intersection of all negative half-spaces L_P^- of supporting hyperplane L_P of the set D. As the set D consists of a finite number of points and so the set M is a polyhedron coinciding with the convex hull of the set D. The boundary of M consists of a finite number of faces $\Gamma_s^{(d)}$ ($\Gamma_s^{(0)}$ is a vertex, where d is its dimension, and s is its number. Thus, the $\Gamma_k^{(0)}$ are vertices, $\Gamma_k^{(1)}$ are edges and so on [2,3]. The dimension of the truncation of $\widehat{f}$ is defined as the dimension of the corresponding face $\Gamma_s^{(d)}$.

We consider $P \in \mathbf{R}_2^n$, $Q \in \mathbf{R}_1^n$, where $\mathbf{R}_1^n$ and $\mathbf{R}_2^n$ are dual spaces.

If the set D is the support of f, then the convex hull M of D is called the Newton polyhedron of f. To each face $\Gamma_s^{(d)}$ of M there corresponds truncation (9) with respect to vectors $P \in N_s^{(d)} = N(f, \widehat{f})$.

2°. We now consider system (1), where the f_i are the finite sums of monomials without similar terms, $D_i \subset \mathbf{Z}^n$. To each f_i there correspond the objects $D_i, M_i, \widehat{f}_i, N(f_i, \widehat{f}_i) \ldots$. First subscript i will show that an object belongs to f_i. System (3) is a truncation of the system (1) for the order P if, for each i sum $\widehat{f}_i$ is a truncation of f_i for the order P.

The cone K of the truncated system (3) is the set of all such P that the truncation of system (1) for order P is the same system (3). It is evident that

$$K = N(f_1, \widehat{f}_1) \cap \ldots \cap N(f_{n-1}, \widehat{f}_{n-1}).$$

Let d_i be the dimension of $\widehat{f}_i$. The dimension d of the truncated system (3) is the codimension of the cone K of the system, i.e., $d = n - \dim K \le d_1 + \cdots + d_{n-1}$.

Theorem 2.1. *Let a curve of the form (10) satisfy system (1). Then the curve*

$$x_i = b_i t^{p_i}, \quad i = 1, \dots, n \tag{11}$$

satisfies the truncated system (3), if P lies in its cone K.

3°. Let us consider the power transformation (4), where the α_{ij} are integers, $\alpha = (\alpha_{ij})$ is a square matrix with $\det \alpha = \pm 1$. The set of such power transformation (4) makes up a group [2]. Denote $\log X = (\log x_1, \dots, \log x_n)$. We shall consider a vector as a matrix column. Then (4) can be written as

$$\log Y = \alpha \log X. \tag{12}$$

It transforms curve (9) into the curve

$$\begin{gathered} y_i = c_i t^{r_i}(1 + o(1)), \quad c_i \neq 0, \quad i = 1, \dots, n \\ \text{where} \quad t \to \infty, \quad R = (r_1, \dots, r_n) \text{ and } R = \alpha P \end{gathered} \tag{13}$$

By (12) we have the chain of equalities

$$X^Q = \exp\langle \log X, Q\rangle = \exp\langle \alpha^{-1} \log Y, Q\rangle = \exp\langle \log Y, \alpha^{*^{-1}} Q\rangle = Y^S,$$

where

$$S = \alpha^{*^{-1}} Q. \tag{14}$$

Hence,

$$f(X) = \sum a_Q X^Q = \sum a_Q Y^S = g(Y); \quad D' = \alpha^{*^{-1}} D.$$

The power transformation (4) induces two linear transformations: $R = \alpha P$ in $\mathbf{R}_1^n$ and (14) in $\mathbf{R}_2^n$. The scalar product is preserved:

$$\langle R, S\rangle = \langle \alpha P, \alpha^{*^{-1}} Q\rangle = \langle P, Q\rangle.$$

So $\mathbf{R}_1^n$ and $\mathbf{R}_2^n$ are dual spaces.

Note that all our constructions commute with power transformations, transformation (4) gives the one-to-one correspondence for the integer lattice $\mathbf{Z}^n$ and (4) is the one-to-one transformation of the torus set

$$\{X : 0 < |x_i| < \infty, \quad i = 1, \dots, n\}.$$

Theorem 3.1. *Let the subsystem of the first k equations of the truncated system (3) have the dimension $e(k)$ for $k = 1, \dots, n-1$. Then there exist integer vectors $T_1, \dots, T_{n-1}$ and an integer unimodular matrix α such that, after the power transformation (4), the first k functions of*

$$\widehat{g}_i(Y) = X^{T_i} \widehat{f}_i(X), \quad i = 1, \dots, n-1 \tag{15}$$

depend on $e(k)$ variables $y_1, \dots, y_{e(k)}$ only (for $k = 1, \dots, n-1$) and supports of all functions

$$g_i(Y) = X^{T_i} f_i(X), \quad i = 1, \dots, n-1$$

lie in the first octant $\{\mathbf{R}^n : R \geq 0\}$.

Hence all functions (15) depend on d variables $y_1, \dots, y_d$ only, where $d = e(n-1)$ is the dimension of the truncated system (3). In many cases Theorem 3.1 gives a possibility to transform a truncated system into a triangular form.

4°. Now consider the basic problem: Let us have system (1), where f_i are the Laurent polynomials, and a convex cone $\mathcal{K}$ in $\mathbf{R}_2^n$, we must find all solutions of system (1) such that in expansion (2) the vector $P \in \mathcal{K}$, $P \neq 0$. The cone $\mathcal{K}$ is called the cone of problem.

The basic problem is said to be reduced if the branches without a coordinate identically equal to zero (or infinity) are to be found.

Obviously, the basic problem breaks up into finitely many reduced problems in which various coordinates x_i are set equal to zero (or infinity).

To solve the reduced problem we form for each f_i the M_i, $\varGamma_{ik}^{(d)}$, $N_{ik}^{(d)}$. Here it suffices to single out all the faces $\varGamma_{ik}^{(d)}$ such that the $N_{ik}^{(d)} \cap \mathcal{K}$ is different from zero. Let $\mathcal{K}_{ik}^{(d)} = N_{ik}^{(d)} \cap \mathcal{K}$ and consider possible nonempty intersections.

$$\mathcal{K}_{1k_1}^{(d_1)} \cap \dots \cap \mathcal{K}_{n-1k_{n-1}}^{(d_{n-1})} \overset{\text{def}}{=} \varPi_\lambda, \quad \lambda = 1, \dots, l.$$

Let $\varPi_\lambda$ be one of these intersections and (3) be the truncated system corresponding to it. If $d_i = 0$ then $\widehat{f}_i = aX^Q$ and $aX^Q = 0$, so one of the coordinates is equal to zero, and they cannot be the solution of the reduced problem. Therefore, in a reduced problem all $d_i > 0$.

We make the power transformation for (3) and the cancellations indicated in Theorem 3.1. Then we have

$$\widehat{g}_i(y_1, \dots, y_d) \overset{\text{def}}{=} X^{T_i} \widehat{f}_i(x_1, \dots, x_n) = 0, \quad i = 1, \dots, n-1 \tag{16}$$

System (1) passes to

$$g_i(y_1, \dots, y_d) \overset{\text{def}}{=} X^{T_i} f_i(x_1, \dots, x_n) = 0, \quad i = 1, \dots, n-1 \tag{17}$$

Now (16) is a truncation of system (17) with cone $\varPi'_\lambda = \alpha \varPi_\lambda \subset \{P : p_1 = \dots = p_d = 0\}$. We must now find the solutions

$$y_i = c_i t^{P_i}(1 + o(1)), \quad i = 1, \dots, n$$

of (17) such that the vector order $P \in \varPi'_\lambda$.

Let us find all solutions $y_i = y_i^0$, $i = 1, \dots, d$, of (16) such that all the $y_i^0 \neq 0$ nor ∞. If there are no such solutions, then the reduced problem does not have solutions (2) with $P \in \varPi_\lambda$. If there are such solutions then we distinguish two cases:

(i) the point $y_1^0, \dots, y_d^0$ is an isolated solution of (16);

(ii) this point lies on some continuous set of the solutions of (16) that is an algebraic set of positive dimension.

Case (i). The truncated system (16) has the form

$$\widehat{g}_i(y_1, \dots, y_d) \overset{\text{def}}{=} g_i(y_1, \dots, y_d, 0, \dots, 0) = 0, \tag{18}$$
$$i = 1, \dots, n-1$$

and its cone

$$\varPi'_\lambda = \alpha\varPi_\lambda \subset \{P : p_1 = \cdots = p_d = 0, p_{d+1} < 0, \ldots, p_n < 0\}.$$

Let $y_1^0, \ldots, y_d^0$ be a solution of system (18). Then

$$Y^0 = (y_1^0, \ldots, y_d^0, 0 \ldots, 0)$$

is a root of (17), by (18). Assume that Y^0 is a simple root of (17), then a single branch corresponds to it, which can be found by the Implicit Function Theorem. Returning to the original variables we obtain the described expansion (2). In this case a branch has been isolated.

But if $Y = Y^0$ is a singular point of (17), or if in (17) the $y_{d+1}, \ldots, y_n$ appear to negative powers, then we make the substitution

$$y_i = y_i^0 + z_i, \quad i = 1, \ldots, d, \quad y_j = z_j, \quad j = d+1, \ldots, n$$

in (7). We obtain the system

$$h_i(z_1, \ldots, z_n) \overset{\mathrm{def}}{=} g_i(y_1, \ldots, y_n) = 0, \quad i = 1, \ldots, n-1 \tag{19}$$

with

$$\varPi''_\lambda = \{P : p_1 < 0, \cdots, p_d < 0, p_{d+1} = \ldots = p_n = 0\} + \varPi'_\lambda,$$

the cone of the problem. For system (19) we have the basic problem, but for a narrower set of solutions.

Case (ii). This case may be considered as degenerate. Here the solutions of system (16) contain a continuum. So to isolate branches we must use not only the first approximation of the system but also the next approximation as well.

The set of solution of an arbitrary system (16) consists of a finite number of the irreducible algebraic manifolds, and the following reduction may be done separately for each of them. Let H be such an l-dimensional manifold $(0 < l < d)$. If it is a linear one, then by a reversible linear change of coordinates $y_1, \ldots, y_d \to z_1, \ldots, z_d$ we transfer H into the coordinate subspace, $z_{l+1} = \cdots, = z_d = 0$ and look for solutions of (16) near this subspace, i.e. we arrive at the basic problem with the cone

$$\mathcal{K} = \{P : p_{l+1} < 0, \cdots, p_d < 0, p_{d+1} = \ldots = p_n = 0\} + \varPi'_\lambda.$$

If the manifold H cannot be transferred into a coordinate subspace then we can compose in some cases from system (17) such an additional equation that its first approximation does not equal zero identically on H, and in other cases we can introduce additional coordinates, which give another truncated system.

The complexity of the truncated system (3) is defined to be the $(n-1)$-dimensional Minkowski [1] mixed volume of the corresponding parallel-transfer faces $\varGamma_{ik}^{(d_i)} - \overline{Q}^i, \quad i = 1, \ldots, n-1$.

The complexity of reduced problem is defined to be the sum of the complexities of all the truncated systems whose normal cones intersect the cone of the problem.

In the generic case the number of complex branches of the reduced problem is equal to its complexity [1]. For a non-generic system of this kind no estimate of the number of branches is known for $n > 2$.

5°. Example

Let the algebraic curve be given by the system of equations

$$\begin{cases} f_1(x) \overset{\text{def}}{=} x_1^4 + 3x_2^4 - 6x_2^2x_3^2 - 9x_2x_3^3 - 6x_3^4 + 7x_1x_2^2x_3^2 = 0, \\ f_2(x) \overset{\text{def}}{=} 2x_1^2 - 4x_2^3 + 3x_3^4 + 2x_1x_2x_3^2 = 0. \end{cases}$$

We distinguish all branches of this curve near the singular point $X = 0$, therefore, the cone of the problem is $K = \{P < 0\}$. We find supports D_1 and D_2:

$$D_1 = \{Q_1^1 = (4;0;0); Q_2^1 = (0;4;0); Q_3^1 = (0;2;2);$$

$$Q_4^1 = (0;1;3); Q_5^1 = (0;0;4); Q_6^1 = (1;2;2)\}$$

$$D_2 = \{Q_1^2 = (2;0;0); Q_2^2 = (0;3;0); Q_3^2 = (0;0;4); Q_4^2 = (1;1;2)\}$$

The Newton polyhedra M_1 and M_2, as the convex hulls of sets D_1 and D_2 are shown in Figures 1 and 2. The results of these computations on PC [4] are shown in Tables 1 and 2. It is immediately clear from Figures 1 and 2 that for M_1 the support planes with $P \le 0$ pass only through the elements of the triangle $\Gamma_{11}^{(2)}$ with vertices Q_1^1, Q_2^1, Q_3^1, and for M_2 they pass through the elements of the triangle $\Gamma_{21}^{(2)}$ with vertices Q_1^2, Q_2^2, Q_3^2.

We write the linear relations (8) on the plane $S = \{p_1 + p_2 + p_3 = -1\}$. For a side $\Gamma_{11}^{(2)}$ and M_1 the relations (8) on this plane we obtain the points $S \cap K_{11}^{(2)} = \{p_2 = p_3 = -\frac{1}{3}\}$ and $S \cap K_{21}^{(2)} = \{p_2 = -\frac{4}{13}, p_3 = -\frac{5}{13}\}$ respectively. the linear relations on a plane S for edges will be:

$$S \cap K_{21}^{(1)} = \{5p_2 + 2p_3 = -2, 3p_2 > 4p_3\};$$

$$S \cap K_{23}^{(1)} = \{3p_3 + p_2 = -1, 4p_3 > 3p_2\};$$

$$S \cap K_{22}^{(1)} = \{3p_2 = 4p_3, 3p_3 + p_2 > -1\};$$

Representing these lines on a plane S we get the picture of the section S of the space $\mathbf{R}^3$. The heavy and light lines in Fig. 3 represent sections of the normal cones $N_{11}^{(1)}$ and $N_{21}^{(1)}$ of the edges of the polyhedra M_1 and M_1, respectively. It is clear from Figure 3 that in the cone of the problem there is only one intersection

$$S \cap \Pi = S \cap K_{21}^{(1)} \cap K_{12}^{(1)} = \left\{p_2 = p_3 = -\frac{2}{7}\right\}$$

Which is a cone of truncation of the following truncated system

$$\left\{\begin{array}{l} \hat{f}_1(x) \overset{\text{def}}{=} 3x_2^4 - 6x_2^2x_3^2 - 9x_2x_3^3 - 6x_3^4 = 0; \\ \hat{f}_2(x) \overset{\text{def}}{=} 2x_1^2 - 4x_2^3 = 0, \end{array}\right\}$$

from here we have

$$\bar{Q}'_1 = \bar{Q}'_2 - \bar{Q}'_5 = (0,1,-1)\,; \bar{Q}^2_2 = \bar{Q}^2_1 - \bar{Q}^2_2 = (2,-3,0)\,.$$

Therefore, the unimodular matrix α is:

$$\alpha = \begin{pmatrix} 0 & 1 & -1 \\ 2 & -3 & 0 \\ 1 & -1 & 0 \end{pmatrix}\,;\ \alpha^{-1} = \begin{pmatrix} 0 & -1 & 3 \\ 0 & -1 & 2 \\ -1 & -1 & 2 \end{pmatrix}.$$

The power transformation with a matrix α and its inverse are:

$$\begin{cases} y_1 = x_2 x_3^{-1}, \\ y_2 = x_1^2 x_2^{-3}, \\ y_3 = x_1 x_2^{-1}, \end{cases} \quad \begin{cases} x_1 = y_2^{-1} y_3^3, \\ x_2 = y_2^{-1} y_3^2, \\ x_3 = y_1^{-1} y_2^{-1} y_3^2. \end{cases}$$

After the power transformation and some cancellation in the truncated system we get the system

$$\begin{cases} y_1^4 - 2y_1^2 - 3y_1 - 2 = 0, \\ y_2 - 2 = 0. \end{cases}$$

This system has two real and two complex simple solutions:

$$y_{11}^0 = -1,\ \ y_2^0 = 2,\ \text{and}\ y_{12}^0 = 2,\ \ y_2^0 = 2.$$

$$y_{13}^0 = \frac{1}{3}\left(-1 + i\sqrt{3}\right),\ y_2^0 = 2,\ \ \text{and}\ \ y_{14}^0 = \frac{1}{2}\left(-1 - i\sqrt{3}\right),\ y_2^0 = 2$$

Making the substitutions $y_1 = -1 + z_1,\ y_2 = 2 + z_2$ and $y_1 = 2 + z,\ y_2 = 2 + z_2$ in the system

$$\begin{cases} f_1\left(y_1, y_2, y_3\right) = 0, \\ f_2\left(y_1, y_2, y_3\right) = 0. \end{cases}$$

applying the implicit functions theorem to the expression obtained, and returning to the original variables with respect to power transformation, we find the real branches $\mathcal{F}_1$, $\mathcal{F}_2$

$$\begin{cases} x_1 = \frac{1}{2} y_3^3 + \frac{3}{16} y_3^5 + o\left(y_3^6\right), \\ x_2 = \frac{1}{2} y_3^2 + \frac{3}{16} y_3^4 + o\left(y_3^5\right), \\ x_3 = -\frac{1}{5} y_3^2 - \frac{3}{16} y_3^4 + o\left(y_3^5\right). \end{cases}$$

Table 1. Table of correspondence of set D_1 (see [4])

	Q_1^1	Q_2^1	Q_3^1	Q_4^1	Q_5^1	Q_6^1
$N_{11}^{(0)} = (-1,-1,-1)$	+	+	+	+	+	-
$N_{12}^{(0)} = (0,-1,0)$	+	-	-	-	+	-
$N_{13}^{(0)} = (-1,0,0)$	-	+	+	+	+	-
$N_{14}^{(0)} = (0,0,-1)$	+	+	-	-	-	-

d	i	j	k
2	1	1,2,3,4,5	1
1	1	1,5	1
1	2	2,3,4,5	1
1	3	1,2	1
0	1	5	1,2
0	2	1	1,3
0	3	2	2,3

Table 2. Table of correspondence of set D_2 (see [4])

	Q_1^2	Q_2^2	Q_3^2	Q_4^2
$N_{21}^{(0)} = (-6, -4, -3)$	+	+	+	-
$N_{22}^{(0)} = (0, -1, 0)$	+	-	+	-
$N_{23}^{(0)} = (-1, 0, 0)$	-	+	+	-
$N_{24}^{(0)} = (0, 0, -1)$	+	+	-	-

d	i	j	k
2	1	1,2,3	1
1	1	1,3	1
1	2	2,3	1
1	3	1,2	1
0	1	3	1,2
0	2	1	1,3
0	3	2	2,3

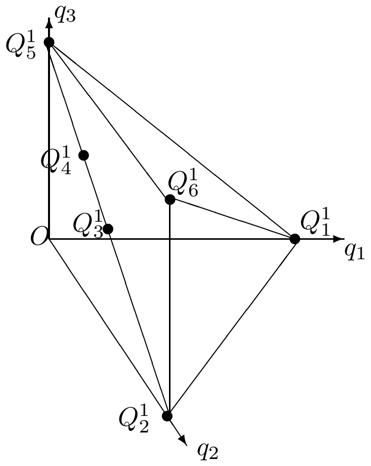


Fig. 1. Polyhedron $\mathbf{M_1}$ of the set $\mathbf{D_1}$

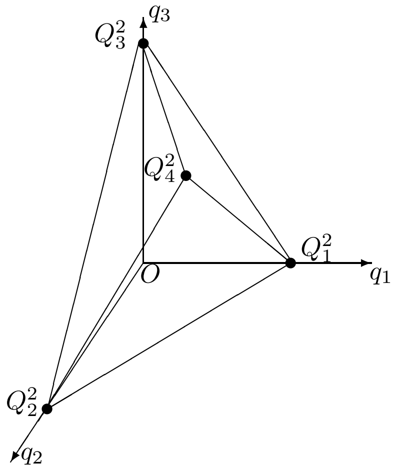


Fig. 2. Polyhedron $\mathbf{M_2}$ of the set $\mathbf{D_2}$

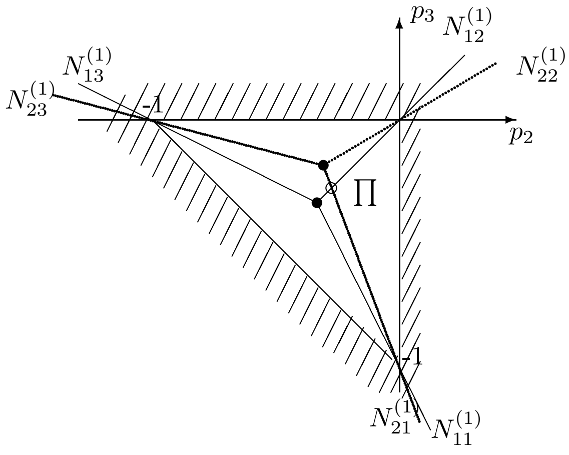


Fig. 3.

$$\begin{cases} x_1 = \frac{1}{2}y_3^3 + \frac{3}{128}y_3^5 + o\left(y^6\right), \\ x_2 = \frac{1}{2}y_3^2 + \frac{3}{128}y_3^4 + o\left(y^5\right), \\ x_3 = \frac{1}{4}y_3^2 + \frac{3}{156}y_3^4 + o\left(y^5\right). \end{cases}$$

The complex roots of the system give the following complex branches $\mathcal{F}_3$, $\mathcal{F}_4$:

$$\begin{cases} x_1 = \frac{1}{2}y_3^3 + \frac{3}{128}y_3^5 + o\left(y^6\right), \\ x_2 = \frac{1}{2}y_3^2 + \frac{3}{128}y_3^4 + o\left(y^5\right), \\ x_3 = -\frac{1}{4}\left(1 \pm i\sqrt{3}\right)y_3^2 - \frac{3}{32}\left(1 \pm i\sqrt{3}\right)y_3^4 + o\left(y^5\right). \end{cases}$$

For the branches found the role of the parameter τ is played by y_3^{-1}.

6°. Computation of branches of solutions of the specific system (1) consists of the following 8 stages:

1. For each coordinate singular point X^0 we do parallel-transfer $X - X^0$, write the system in the form (1) and make following stages for each such system separately. We shall describe them for system (1).

2. For each f_i compute the Newton polyhedron M_i, all its faces $\Gamma_{ij}^{(d_i)}$, and normal cones $N_{ij}^{(d_i)}$ and sets $D_{ij}^{(d_i)}$.

3. Find all nonempty intersections $N_{1j}^{(d_1)} \cap \ldots \cap N_{n-1j}^{(d_{n-1})}$ with all $d_i > 0$ and, for each of them, write the corresponding truncated system (3).

4. For each such system (3), compute vectors T_i and the matrix α by Theorem 3.1 and make corresponding transformations of (1) and (3) into (17) and (16).

5. Find all roots of (6) and, by computation of the matrix $G = (\partial g_i/\partial y_j)$ separate simple roots Y^0 of (17).

6. By Implicit Function Theorem, compute an initial part of expansions for the branch corresponding to the simple root Y^0 of (17).

7. For each non-simple root Y^0 of (17), compute the new system (19) and repeat the procedure until a full isolation of all branches.

8. By inverse transformations, write all branches in initial coordinates X.

Stages 1–4 were programmed in PC. Stages 5, 6, and 8 are essentially nonlinear but can be done by standard programs.

References

1. Bernshtein, D.N.: The number of roots of a system of equatios, Funct. Anal. Appl. 9 (1975)
2. Bruno, A.D.: Power Geometry for Algebraic and Differential Equations, Elsevier, Amsterdam (2000)
3. Soleev, A.: Singling out branches of an algebraic curve and Newton polyhedra, Dokl. Akad. Nauk SSSR 268 (1983), 1305–1307; English transl. in Soviet Math Dokl. 27 (1983)
4. Soleev, A. and Aranson, A.: Computation of a polyhedron and of normal cones of its faces. Preprint No. 4 of the Keldysh Institute of Applied Mathematics of RAS, Moscow (1994)
5. Bruno, A.D. and Soleev, A.: Local uniformization of branches of a space curve, and Newton polyhedra. St. Peterburg Math. J. 3:1 (1992) 53–82

6. Varchenko, A.N. and Khovanskii, A.G.: Asymptotics of integrals over vanishing cycle and the Newton polyhedron, Dokl. Akad. Nauk SSSR 283 (1985), 211–262; English transl. in Soviet Math Dokl. 32 (1985)
7. Khovanskii, A.G.: Newton polyhedra(resolution of singularities), Itogi Nauki i Tekhniki: Sovremennye Problemy Mat., Vol. 22, VINITI, Moscow, (1983), pp. 207-239; English transl. in J. soviet Math 27 (1984)
8. Chernikov, S.N.: Linear Inequalities, Nauka, Moscow (1968); German transl., VEB Deutcsher Verlag, Berlin (1971)
9. Graves, L.M.: Remarks on singular points of functional equations, Trans. Amer. Math. Soc. 79 (1955), 150-157
10. Vainberg, M.M., Trenogin, V.A.: The theory of Branching of Solutions of Nonlinear Equations. Wolters-Noordholf Intern. Publ., Leiden (1974)
11. Botashev, A.I.: Analytic methods in the theory of branching, Uspekhi Mat. Nauk 40 (1985), no. 4 (244), 147–148; English transl. in Russian Math. Surveys 40 (1985)

Quantifier Elimination for Constraint Logic Programming

Thomas Sturm

FMI, Universität Passau, 94030 Passau, Germany
sturm@uni-passau.de
www.uni-passau.de/~sturm

Abstract. We present an extension of constraint logic programming, where the admissible constraints are arbitrary first-order formulas over various domains: real numbers with ordering, linear constraints over p-adic numbers, complex numbers, linear constraints over the integers with ordering and congruences (parametric Presburger Arithmetic), quantified propositional calculus (parametric QSAT), term algebras. Our arithmetic is always exact. For $\mathbb{R}$ are $\mathbb{C}$ there are no restrictions on the polynomial degree of admissible constraints. Constraint solving is realized by effective quantifier elimination. We have implemented our methods in our system CLP(RL). A number of computation examples with CLP(RL) are given in order to illustrate the conceptual generalizations provided by our approach and to demonstrate its feasibility.

1 Introduction

During the past 15 years, quantifier elimination has become a successful and universal tool in various fields of science and engineering. An overview on some relevant application areas has been given in [1], which in turn give numerous references to further literature on actual as well as on possible applications.

According to the author's experience there are three major points stated by the numerous researchers from outside the core quantifier elimination community, who are interested in applying such methods:

1. The request for further domains of computation as e.g. the integers, mixed real-integer domains, or quantified propositional calculus. Interestingly, these requests restrict mostly to theories that actually admit quantifier elimination, such that these issues are mainly a matter of time and of man power.
2. The combination of various domains for modeling a problem. The requested combinations in general clearly exceed the applicability of pure quantifier elimination approaches. One important exception is the mixed real-integer quantifier elimination proposed in [2].
3. As a further extension of point 2 above, there is often a need for free terms and predicates in the style of logic programming.

The two latter points give rise to the idea that one or several quantifier elimination procedures can be combined with resolution techniques. Instead of developing such a concept from scratch, we observe that quantifier elimination can perfectly take the role of a constraint solver in constraint logic programming (CLP). This paper is going to describe how quantifier elimination can be integrated into this framework. Hence our work is going to extend CLP by admitting as constraints arbitrary *first-order* formulas.

In a pure CLP approach, one would fix a domain, and then all variables of a program would be considered variables over this domain. Then the constraint solver completely replaces unification also for non-constraint subgoals. A more general approach would admit several types of variables: variables over the domain of the solver, variables over other domains treated by other solvers, and general variables to be traditionally treated by unification. All these extensions do not at all interfere with the issues that we are going to discuss here. We will thus allow ourselves to restrict to the basic case that all variables are elements of the domain of our solver, and that this solver is the only one involved.

The future applications that we have in mind are not those currently handled with constraint solving systems. Instead we focus on problems currently solved by quantifier elimination with human interaction. Hopefully, it will in addition be possible to fill a considerable part of the gap between the paradigms of constraint logic programming on the one hand and quantifier elimination on the other hand. So although talking about extending constraint solving systems throughout this article, the intuition is primarily that the major part of the job is done by the quantifier elimination engine, while there is a bit of logic programming on top of this.

If our approach would also be noticed and picked up by the constraint solving community this might be, of course, mutually fruitful. There is in fact also a nice theoretical benefit of our approach from the point of view of logic programming: With the recent developments in quantifier elimination techniques for term algebras [3], the unification for non-constraint terms within the resolution processes can in principle be replaced by quantifier elimination. This has the consequence that the free terms formally establish just another domain and do not play a special role anymore.

We want to mention some existing work that might appear similar to our ideas at the first glance: First, Hong has described a CLP system RISC-CLP(REAL) [4]. It admits real constraints of arbitrary degree. They are solved by either real quantifier elimination or by Gröbner basis methods. This approach, however, has the usual restriction of pure lists of constraints in contrast to arbitrary first-order formulas.

Second, there is considerable research being done on *first-order constraints.* This so far mostly affects the treatment of finite domains, finite or infinite terms, approximate methods, and decidability considerations for very general domains. The connection between researchers developing such first-order methods on the one hand, and researchers seeking such methods for incorporation into their CLP framework on the other hand appears to be quite weak [5].

The plan of this paper is as follows: In Section 2, we introduce the notion of quantifier elimination. We also give an overview on the currently available domains for this: $\mathbb{R}$, $\mathbb{Q}_p$ (linear theory of p-adic numbers), $\mathbb{C}$, $\mathbb{Z}$ (parametric Presburger Arithmetic), quantified propositional calculus (parametric QSAT), term algebras. In Section 3 we give an analogous introduction to the theoretical background of CLP to the extent that is necessary for our purposes here. Section 4 describes in detail our CLP interpreter by giving an explicit algorithm. In addition we indicate an alternative point of view on our work, which would not revise present CLP interpreters but instead access our quantifier elimination techniques as *reasoning services*. Section 5 gives a brief overview of our implementation. This consists of a system CLP(RL), which is embedded in the computer algebra system REDUCE. There all necessary quantifier elimination methods are provided by the computer logic package REDLOG [6].

Section 6 collects some computation examples with CLP(RL). This gives an idea of the present efficiency of our system. In addition, these examples illustrate to what extent our approach conceptually extends the traditional framework of CLP:

– constraints of arbitrary degree,
– exact real arithmetic for arbitrary degree,
– absolutely complete treatment of e.g. disjunction,
– quantified constraints,
– new powerful domains.

In Section 7 we finally summarize our results and evaluate our work.

2 Quantifier Elimination

Quantifier elimination generally takes place wrt. a fixed formal language and a semantics for that language. For our purposes here, it is convenient to call such combinations *domains*. We start with a brief survey of the various domains that we provide at present. On this basis we then turn to the notion of quantifier elimination.

2.1 Available Domains

real numbers. We use the language $\mathcal{L}_{\mathrm{OR}} = (0, 1, +, -, \cdot\, ; \leqslant, \geqslant, <, >, \neq)$ of ordered rings expanded by constants for all rational numbers; formally that is $\mathcal{L}_{\mathrm{OR}}(\mathbb{Q})$. Available implemented quantifier elimination methods are the following: *partial cylindrical algebraic decomposition* (CAD) [7]; *virtual substitution methods* [8,9]; *Hermitian quantifier elimination* [10,11,12]. All these methods have certain advantages and disadvantages. For a thorough survey we point the reader at [1]. Within the framework described here, the users need not care about these issues. Appropriate methods will be chosen automatically via some heuristic approach.

p-adic numbers. We use the language $\mathcal{L}_{\mathrm{DIV}} = (0, 1, p, +, -, \cdot\, ; \mid, \parallel, \sim, \not\sim, \neq)$ of rings with abstract divisibilities expanded by constants for all rational numbers; formally $\mathcal{L}_{\mathrm{DIV}}(\mathbb{Q})$. Using abstract divisibilities in contrast to a function symbol v for the valuation allows us to keep things one-sorted. The divisibility predicates are defined as follows:

$$a \mid b :\Leftrightarrow v_p(a) \leqslant v_p(b), \quad a \parallel b :\Leftrightarrow v_p(a) < v_p(b), \quad a \sim b :\Leftrightarrow v_p(a) = v_p(b).$$

Our implemented methods are restricted to the corresponding *linear theory.* That is, the total degree in the quantified variables must not exceed 1. This restriction is a bit weakened by the automatic application of various heuristic methods, e.g. polynomial factorization. Our quantifier elimination procedure by *virtual substitution* [13] allows to keep the prime p parametric. For our purposes here, we have to fix p, however. This guarantees that variable-free atomic formulas can be decided, and on these grounds arbitrary sentences can be decided by employing quantifier elimination.

complex numbers. We use the language $\mathcal{L}_{\mathrm{R}} = (0, 1, +, -, \cdot\,; \neq)$ of rings expanded by constants for all rational numbers; formally $\mathcal{L}_{\mathrm{R}}(\mathbb{Q})$. Notice that there are no operations like real part, imaginary part, or absolute value available. Problems involving such operations would be encoded as real problems instead. Our quantifier elimination methods are based on *comprehensive Gröbner bases* [14].

integers. We use the language $\mathcal{L}_{\mathrm{PR}} = (0, 1, +, -, \cdot\,; \leqslant, \geqslant, <, >, \neq, \equiv^{(3)}, \not\equiv^{(3)})$ of ordered rings expanded by (ternary) congruences and their negated counterparts plus constants for all integers; formally $\mathcal{L}_{\mathrm{PR}}(\mathbb{Z})$. The theory is the linear theory of the integers over the above language. This is commonly known as *Presburger Arithmetic.* Our framework is actually a bit more general: *linearity* refers to the total degree of the quantified variables only. As coefficients and also as moduli there may occur arbitrary polynomials in the parameters, i.e., in the unquantified variables [15].

quantified propositional calculus. We formally use the essentially algebraic language $\big(0^{(0)}, 1^{(0)}, \sim^{(1)}, \&^{(2)}, |^{(2)}, \rightarrow^{(2)}, \leftrightarrow^{(2)}; \neq\big)$, where all symbols except the negated equality "$\neq$" are function symbols. Over this language, we consider the theory of initial, i.e. 2-element, Boolean algebras. Quantifier elimination is performed by *virtual substitution* in combination with sophisticated intermediate simplification techniques in the style of non-probabilistic approaches to SAT-checking. Note that in terms of that framework, quantifier elimination offers *parametric* QSAT-*checking.* There are normal forms of formulas such that all atomic formulas are of one of the forms $v = 0$ or $v = 1$, where v is a variable. Using a so-called *propositional wrapper* such formulas can be straightforwardly displayed in the style of quantified propositional calculus. See [16] for further details on this domain.

term algebras. over finite languages. This domain is motivated by the desired treatment of *absolutely free* term algebras over finite algebraic languages $\mathcal{L}$. Except for extreme special cases, such structures do not admit quantifier elimination. This is overcome by expanding the languages by symbols for unary so-called *inverse functions.* These inverse functions are essentially used to express whether their argument term t starts with a particular function symbol from $\mathcal{L}$ or not. See [3] for further details on this domain and for our quantifier elimination method by virtual substitution.

2.2 The Notion of Quantifier Elimination

As already indicated, a domain is determined by two choices: a language and a corresponding semantics.

We have explicitly given the languages for our domains in the previous section. In order to give a formal framework for real quantifier elimination, we introduce *first-order logic* on top of such languages $\mathcal{L}$. *Terms* are inductively defined by nesting according to their arities function symbols and constants and also *variables*, which are not mentioned in the languages. Many of our languages are expansions of the language of rings. In these cases, every term can be equivalently represented by a multivariate polynomial f with rational coefficients.

Atomic formulas are either equations $t = t'$ between terms t, t' or *predicates.* A predicate is built by combining a relation symbol $R^{(n)}$ from $\mathcal{L}$ with terms t_1, ..., t_n according to its arity n; this yields a word $R(t_1, \ldots, t_n)$. Wherever this is appropriate, we make use of infix notation in both our description here and our implementations. Similarly to the choice of polynomials as canonical term representations above, there are often reasonable normal forms for atomic formulas. Over the reals, e.g., it is sufficient to consider equations and inequalities of the form $f = 0$, $f \leqslant 0$, $f \geqslant 0$, $f < 0$, $f > 0$, or $f \neq 0$ for polynomials f.

Quantifier-free formulas are "true," "false," atomic formulas, and any combination between these by the logical operators "$\neg$," "$\wedge$," "$\vee$," "$\longrightarrow$," "$\longleftarrow$," "$\longleftrightarrow$."

A formula of the form $\exists x_1 \ldots \exists x_n \psi(u, x)$, where $\psi(u, x)$ is a quantifier-free formula, is called an *existential formula.* Similarly, *universal formulas* are of the form $\forall x_1 \ldots \forall x_n \psi(u, x)$. A *prenex first-order formula* has several alternating blocks of existential and universal quantifiers in front of a quantifier-free formula. General *first-order* formulas are inductively obtained from "true," "false," and atomic formulas by admitting quantification also inside the scope of logical operators. It is not hard to see, however, that every first-order formula is equivalent to a prenex one. Notice our convention to denote *parameters* by $u = (u_1, \ldots, u_m)$ and *main variables* by $x = (x_1, \ldots, x_n)$. A parameter-free first-order formula is called a first-order *sentence.* A first-order formula $\varphi(u)$ (possibly containing quantified main variables x) can be turned into a sentence by universally quantifying all present parameters u. The obtained sentence $\underline{\forall}\varphi = \forall u_1 \ldots \forall u_m \varphi(u)$ is called the *universal closure* of φ.

For the semantics of our domains, we had used in the previous section verbal descriptions like "the real numbers." Let us be a bit more precise about that now. Let $\mathbb{S}$ be the $\mathcal{L}$-structure that we are aiming at, e.g. $\mathbb{S} = \mathbb{R}$. Then the *theory* $\mathrm{Th}(\mathbb{S})$ of $\mathbb{S}$ is the set of all first-order sentences that hold over $\mathbb{S}$. Note that $\mathrm{Th}(\mathbb{S})$ depends on $\mathcal{L}$ although the notation does not reflect this. If for some first-order formula φ its universal closure $\underline{\forall}\varphi$ is contained in $\mathrm{Th}(\mathbb{S})$, then we say that φ *holds in the theory of* $\mathbb{S}$ or shorter φ *holds in* $\mathbb{S}$; we write $\mathrm{Th}(\mathbb{S}) \models \varphi$ or $\mathbb{S} \models \varphi$, respectively. Any set Φ of first-order formulas naturally induces a *model class*, i.e. the class of all $\mathcal{L}$-structures in which all formulas from Φ hold: $\mathrm{Mod}(\Phi) = \{\, \mathbb{T} \mid \mathbb{T} \models \varphi \text{ for all } \varphi \in \Phi \,\}$. For arbitrary classes $\mathfrak{M}$ of $\mathcal{L}$-structures and first-order formulas φ we write $\mathfrak{M} \models \varphi$ if $\mathbb{T} \models \varphi$ for all $\mathbb{T} \in \mathfrak{M}$. For instance,

$\mathrm{Mod}(\mathrm{Th}(\mathbb{R}))$ is the class of real closed fields, and $\mathrm{Mod}(\mathrm{Th}(\mathbb{R})) \models x^2 \geqslant 0$, in particular $\mathbb{R} \models x^2 \geqslant 0$.

The *quantifier elimination problem* can be phrased as follows: Given a formula φ, find a quantifier-free formula φ' such that φ' is equivalent to φ in the considered domain; formally $\mathbb{S} \models \varphi' \longleftrightarrow \varphi$. A procedure computing such a φ' from φ is called a *quantifier elimination procedure.* Note that, since the required equivalence is a first-order formula itself, the quantifier elimination output φ' generally satisfies even $\mathrm{Mod}(\mathrm{Th}(\mathbb{S})) \models \varphi' \longleftrightarrow \varphi$. As a simple example for real quantifier elimination consider the equivalence $\mathbb{R} \models a \neq 0 \lor b = 0 \longleftrightarrow \exists x(ax + b = 0)$.

Over the reals, there is a straightforward geometric interpretation for quantifier elimination for an existential formula $\varphi(u) = \exists x_1 \ldots \exists x_n \psi(u, x)$. Consider $M = \{\, (u, x) \in \mathbb{R}^{m+n} \mid \psi(u, x) \,\}$ and $M' = \{\, u \in \mathbb{R}^m \mid \varphi(u) \,\}$. Then M' is the projection of M along the coordinate axes corresponding to the existentially quantified variables x into the parameter space. Real quantifier elimination yields a quantifier-free description of this projection.

3 Constraint Logic Programming

Constraint logic programming (CLP) arose around the mid of the eighties. Prominent systems are CHIP [17], CLP(R) [18], and Prolog III [19]. CLP integrates the concepts of logical programming with constraint solving. The intuitive idea is that a constraint is a relational dependence between several numbers (in a very liberal sense), variables, and certain functions on these numbers and variables. The type of numbers and the possible functions and relational dependences establish the domain of the constraint solver. Our domains introduced in Section 2.1 are compatible with this notion. In fact, they are quite typical examples. A *solution* of a constraint system is *one* binding of all involved variables such that all constraints are simultaneously satisfied. A constraint solver computes such a solution. In particular, it checks this way for feasibility, i.e. the existence of a solution. We now turn this informal discussion into generalized formal definitions suitable for our framework.

We fix a domain $\mathbb{S}$ with language $\mathcal{L}$. A *constraint* is a first-order formula in the sense of Section 2.1. An *atom* is of the form $P(t_1, \ldots, t_n)$ for $n \in \mathbb{N}$ where P is an n-ary predicate symbol that is not in $\mathcal{L}$, and the $t_1, \ldots, t_n$ are terms. Atoms must not be confused with atomic formulas.

A *clause* is of the form $\beta_0 \leftarrow \beta_1, \ldots, \beta_n, \psi$, where $\beta_0, \ldots, \beta_n$ are atoms, and ψ is a constraint. The atom β_0 is the *head* of the clause. The sequence $\beta_1, \ldots, \beta_n, \psi$ is the *body* of the clause. Notice that it is not possible to have a constraint as the head of a clause. A *program* is a finite set of clauses. A *query* is of the form $\leftarrow \alpha_1, \ldots, \alpha_n, \varphi$, where $\alpha_1, \ldots, \alpha_n$ are atoms, and φ is a constraint.

Let Π be a program, and let Q be a query. Then we can fix an expansion language $\mathcal{L}' \supseteq \mathcal{L}$ containing all predicate symbols and function symbols occurring in Π and Q. Over this language $\mathcal{L}'$ we identify clauses $\beta_0 \leftarrow \beta_1, \ldots, \beta_n, \psi$ with first-order formulas $\beta_0 \longleftarrow \beta_1 \land \cdots \land \beta_n \land \psi$. Accordingly, we identify the program Π with the conjunction $\bigwedge \Pi$ of the contained clauses. Finally, the empty head in the query Q is interpreted as “false,” and thus $Q = \leftarrow \alpha_1, \dots, \alpha_n, \varphi$ is equivalent to $\neg(\alpha_1 \wedge \dots \wedge \alpha_n \wedge \varphi)$. Recall that “true” is a constraint, and that forming conjunctions is a valid operation for constructing constraints. It is thus not a restriction that clauses and queries contain exactly one constraint.

Let Π be a program. The *completion* $\bar{\Pi}$ of Π is obtained by adding to Π for each n-ary predicate symbol P in Π a first-order formula γ_P as follows: Let P be defined by clauses

$$\begin{gathered} P(t_{11}, \dots, t_{1n}) \leftarrow B_1 \\ \vdots \\ P(t_{m1}, \dots, t_{mn}) \leftarrow B_m. \end{gathered}$$

Let x_1, …, x_n be pairwise distinct variables not occurring in these clauses. Denote for $i \in \{1, \dots, m\}$ by y_{i1}, …, y_{ik_i} the variables occurring in the i-th clause above. Then γ_P is defined as

$$P(x_1, \dots, x_n) \longleftrightarrow \bigvee_{i=1}^{m} \exists y_{i1} \dots \exists y_{ik_i} \Big(\bigwedge_{j=1}^{n} x_i = t_{ij} \wedge B_i \Big).$$

Note that in the special case that P does not occur in the head of a clause, this amounts to $P(x_1, \dots, x_n) \longleftrightarrow$ false, which is equivalent to $\neg P(x_1, \dots, x_n)$.

A variable v is *free* in a first-order formula φ if v occurs in φ at some point outside the scope of all quantifiers $\exists v$ and $\forall v$. We denote by $\operatorname{var}(\varphi)$ the set of all variables that are free in φ. Let $V = \{v_1, \dots, v_k\}$ be a finite set of variables. Then $\exists V \varphi$ is a concise notation for $\exists v_1 \dots \exists v_k \varphi$. Even more concisely, $\underline{\exists}\varphi$ stands for the *existential closure* $\exists \operatorname{var}(\varphi)\, \varphi$. Accordingly, the universal closure $\underline{\forall}\varphi$, which we have introduced already in Section 2.2, equals $\forall \operatorname{var}(\varphi)\, \varphi$. In view of our discussed identification of syntactic entities of CLP programs with first-order formulas, all these definitions can obviously be applied also to the former.

Let Π be a program, and let $Q = \leftarrow \alpha_1, \dots, \alpha_k, \varphi$ be a query. Then a *correct answer* for Π and Q is a quantifier-free constraint φ' such that $\operatorname{var}(\varphi') = \operatorname{var}(Q)$ and

$$\operatorname{Mod}\big(\bar{\Pi} \cup \operatorname{Th}(\mathbb{S})\big) \models \varphi' \longrightarrow \alpha_1 \wedge \dots \wedge \alpha_k \wedge \varphi.$$

Here the model class is formed over the expanded language $\mathcal{L}'$ while $\operatorname{Th}(\mathbb{S})$ denotes the $\mathcal{L}$-theory of $\mathbb{S}$.

4 Our Resolution Algorithm

We are now going to precisely describe our resolution algorithm in a procedural style. Later on in this section, we discuss an alternative point of view on our work, which would not revise present CLP interpreters but instead access our quantifier elimination techniques as reasoning services there.

Besides quantifier elimination, our following resolution algorithm applies a *simplifier* to the final result formula. Simplifiers map first-order formulas to simpler equivalent ones. For details and a discussion of the notion of simplicity see [20]. There are powerful simplifiers available for all our domains. Generally quantifier elimination implementations would include such simplifiers either as a subroutine for the simplification of intermediate results or as a part of the quantifier elimination procedure itself.

Algorithm 1. *Input: A program Π and a query $Q = \leftarrow \alpha_1, \dots, \alpha_k, \varphi$. Output: A correct answer for Π and Q. Termination: The algorithm does not necessarily terminate. Used subroutines:* simplify *is a simplifier;* qe *is a quantifier elimination procedure.*

```
begin
    $(G', C') := \mathrm{clpqe}\big(\{\alpha_1, \dots, \alpha_k\}, \varphi\big)$
    $C' := \mathrm{simplify}(C')$
    return $C'$
end

procedure clpqe$(G, C)$
begin
    $V := \mathrm{var}(C) \setminus (\mathrm{var}(G) \cup \mathrm{var}(Q))$
    $C := \mathrm{qe}(\exists V C)$
    if $G = \varnothing$ or $C = \mathrm{false}$ then
        return $(G, C)$
    fi
    $G :=$ remove $P(t_1, \dots, t_n) \in G$ from $G$
    $\Pi' :=$ standardize apart all variables in $\Pi$
    while $\Pi' \neq \varnothing$ do
        $\Pi' :=$ remove a clause $K \in \Pi'$ from $\Pi'$
        if $K$ is of the form $P(s_1, \dots, s_n) \leftarrow B, \psi$ then
            $\mu := \bigwedge_{i=1}^{n} s_i = t_i$
            if $\mathrm{qe}(\underline{\exists}(\mu \wedge C)) = \mathrm{true}$ then
                $(G', C') := \mathrm{clpqe}(G \cup B, C \wedge \mu \wedge \psi)$
                if $G' = \varnothing$ and $C' \neq \mathrm{false}$ then
                    return $(G', C')$
                fi
            fi
        fi
    od
    return $(G, \mathrm{false})$
end
```

Consider the domain of the real numbers. Given the program

$$\Pi = \big\{p(X) \leftarrow X \geqslant 0,\, p(X) \leftarrow X \leqslant 0\big\}$$

and the query $\leftarrow p(X)$, it is clear that "true" is a correct answer. There are, however, exactly two possible answers that can be computed by the algorithm: $X \geqslant 0$ and $X \leqslant 0$. This is not a drawback of our particular approach, but a general problem of CLP in contrast to conventional logic programming. CLP resolution is complete only in the following sense:

Theorem 2 (Completeness of CLP). *Let Π be a program, let Q be a query, and let φ' be a correct answer for Π and Q. Then there are finitely many runs of Algorithm 1 on Π and Q with corresponding results $\varphi'_1, \ldots, \varphi'_r$ such that*

$$\mathbb{S} \models \varphi' \longrightarrow \bigvee_{i=1}^{r} \varphi'_i. \quad \square$$

A proof for this restricted completeness as well as for the correctness of Algorithm 1 can be derived from the corresponding proofs for other constraint solvers in any textbook on CLP.

Our current implementation resolves the non-determinism in Algorithm 1 exclusively by selecting the first possible program clause in the order of notation and maintaining a stack of goals. Though common in logic programming, this is a considerable restriction, because it can lead to infinite runs of the algorithm where there would exist finite runs yielding correct answers.

Since the availability of our implementation is an important part of the work discussed here, we have discussed the implemented algorithm to some detail. Our approach is, however, much more flexible than providing this very algorithm: Within the CLP framework, the constraint solver is generally considered as a module that abstractly provides certain *reasoning services*, which are roughly the following [21]:

1. consistency test (is a given constraint satisfiable?),
2. projection (existential elimination of variables),
3. entailment test (decision of implication between constraints),
4. simplification (replace a constraint by an equivalent simpler one),
5. determination (is the value of a variable uniquely determined?),
6. negation (provide suitable constraint symbols to resolve logical negation).

It is not hard to see that even admitting first-order constraints, the services 1–3 can be rather straightforwardly realized by quantifier elimination. Re 4, powerful simplifiers lie at the very heart of any efficient implementation of quantifier elimination. So one can count on their availability. Re 5, the question whether a variable is uniquely determined is again a quantifier elimination problem. Determining the unique value is rather simplification. Note that it is not generally possible to explicitly express all possible unique values anyway. Re 6, finally, the fact that it is useful to encode negation in the language of the domain is well-known also within the quantifier elimination community. Consequently all our domains in Section 2.1 are designed that way.

5 Implementation: CLP(RL)

Our system CLP(RL) is implemented within the computer algebra system REDUCE on top of our computer logic package REDLOG. REDLOG is an extension of the computer algebra system REDUCE to a computer logic system, which provides symbolic algorithms on first-order formulas wrt. temporarily fixed domains in the sense of Section 2.1. REDLOG has its origin in the implementation of real quantifier elimination by virtual substitution. Successfully applying such methods to both academic and real-world problems, the authors have developed a large collection of formula-manipulating tools, many of which are meanwhile interesting in their own right.

The author started the realization of REDLOG in 1992. In April 1995, the system has been redesigned by the author together with A. Dolzmann [6]; in 2001 A. Seidl has joined the development team. In October 1996, REDLOG 1.0 was published on the Web. In April 1999, REDLOG 2.0 was distributed as a part of REDUCE 3.7. In April 2004, the current version REDLOG 3.0 has been shipped with REDUCE 3.8. For many years, REDLOG is widely accepted inside the scientific community. For a collection of applications of REDLOG in various fields of science and engineering see [1] and the references there.

In the name CLP(RL), the "RL" is an abbreviation for REDLOG. Similarly to REDLOG, our present version of CLP(RL) is completely integrated into REDUCE. There is a function `clp`, which has two arguments: a program and a query. A program is a list of clauses; clauses are input following the usual conventions writing "$\longleftarrow$" as "`:-`". The return value of `clp` is a quantifier-free formula in the query variables, which can then be further processed by REDUCE, REDLOG, or CLP(RL) itself. The availability of the computer algebra environment is particularly interesting since for principal reasons the answer formula does not necessarily provide an explicit description of the solution.

The function `clp` is essentially an implementation of Algorithm 1 with the selection rules for goals and clauses as discussed in Section 4. In order to be as clear and concise as possible, we have in our presentation of Algorithm 1 in this paper neglected some efficiency considerations, which we have taken care of in the implementation.

The domain is determined via the global domain choice mechanism of REDLOG. At the present stage of the implementation it is not yet possible to integrate several domains within one program.

6 Computation Examples

We give some computation examples. The idea is at the first place to point at features of our systems that exceed those of many other existing ones. All our computation times mentioned below refer to computations on an Intel 933 MHz Pentium III using 128 MB RAM.

6.1 Real Constraints of Arbitrary Degree

As an example for quadratic real constraints, consider the computation of Pythagorean triples, i.e., natural numbers x, y, $z \in \mathbb{N}$ with $x^2 + y^2 = z^2$. We use the following program:

$$\begin{aligned} \mathrm{nat}(0) &\leftarrow \\ \mathrm{nat}(X+1) &\leftarrow \mathrm{nat}(X),\ X \geqslant 0 \\ \mathrm{pyth}(X,Y,Z) &\leftarrow \mathrm{nat}(X),\ \mathrm{nat}(Y),\ \mathrm{nat}(Z),\ 2 \leqslant X \leqslant Y \leqslant Z \wedge X^2+Y^2=Z^2. \end{aligned}$$

The query $\leftarrow \mathrm{pyth}(3,4,Z)$ yields $Z-5=0$ in 0.05 s. For $\leftarrow \mathrm{pyth}(X,9,Z)$ we obtain $X-12=0 \wedge Z-15=0$ in 0.7 s, and $\leftarrow \mathrm{pyth}(X,Y,9)$ results in "false" after 0.3 s.

A completely parametric query $\leftarrow \mathrm{pyth}(X,Y,Z)$ would result in an infinite run. The reason for this is our fixed clause selection function and thus a general problem of logic programming, which is not really related to our work here.

Our next example has been introduced by Hong for the illustration of his RISC-CLP(REAL) system [4]. The program describes the Wilkinson polynomial equation:

$$\mathrm{wilkinson}(X,E) \leftarrow \prod_{i=1}^{20}(X+i) + EX^{19} = 0.$$

Mind that this product actually occurs in the program in the following expanded polynomial form:

$$\begin{aligned} X^{20} &+ (210+E)X^{19} + 20615X^{18} + 1256850X^{17} + 53327946X^{16} + 1672280820X^{15} \\ &+ 40171771630X^{14} + 756111184500X^{13} + 11310276995381X^{12} \\ &+ 135585182899530X^{11} + 1307535010540395X^{10} + 10142299865511450X^{9} \\ &+ 63030812099294896X^{8} + 311333643161390640X^{7} + 1206647803780373360X^{6} \\ &+ 3599979517947607200X^{5} + 8037811822645051776X^{4} \\ &+ 12870931245150988800X^{3} + 13803759753640704000X^{2} \\ &+ 8752948036761600000X + 2432902008176640000. \end{aligned}$$

On the query $\leftarrow \mathrm{wilkinson}(X,0),\ -20 \leqslant X \leqslant -10$ we obtain after 0.3 s the answer

$$\bigvee_{i=1}^{20} X+i=0.$$

For the query $\leftarrow \mathrm{wilkinson}(X,2^{-23}),\ -20 \leqslant X \leqslant -10$ with a slight perturbation, we obtain after 0.9 s the following answer (in expanded form):

$$8388608 \cdot \left(\prod_{i=1}^{20}(X+i) + 2^{-23}X^{19}\right) = 0 \wedge X+20 \geqslant 0 \wedge X+10 \leqslant 0.$$

The integer factor is the least common denominator of the coefficients of the product polynomial. This answer is contradictory. This could be tested, e.g., by applying quantifier elimination to its existential closure. Hong's RISC-CLP(REAL) actually delivers the result "false." It generally applies some sophisticated processing to its results including DNF computation at the risk of exponentially increasing the size of the output. Since our CLP(RL) lives inside a computer algebra system, we prefer to leave to the user the responsibility of how to proceed. In this situation it would be straightforward to apply the partly numerical function `realroots` [22] of REDUCE to the left hand side polynomial of the equations. This yields after 0.5 s the result

$$X \in \{-20.8469, -8.91725, -8.00727, -6.9997, -6.00001, -5, -4, -3, -2, -1\}.$$

If one wishes to remain exact, one could, e.g., apply CAD to the existential closure of the answer, which immediately yields "false." In general, numerical methods will be more efficient, of course.

6.2 Exact Arithmetic

The minimal perturbation of Wilkinson's equation in the previous section has dramatically demonstrated how sensitive the root behavior of polynomials and thus algebraic equations and inequalities are even to smallest rounding errors. Within CLP(RL) all arithmetic is exact. The price for this is that we possibly obtain only implicit solutions as we have also seen in the previous section. Then one has the choice to either remain exact, or to apply approximate methods.

As long as we are within the CLP framework, however, we remain absolutely exact, and the answers—though not necessarily explicit—are always of the best possible quality from the point of view of exactness.

6.3 Disjunction

Recall that in traditional CLP, constraints are finite sets of relational dependences, which are regarded as conjunctions. There has been a considerable discussion within the CLP community about disjunctions of constraints within clauses and corresponding modifications of the resolution algorithm for treating certain restricted variants of disjunction in an appropriate way. All suggested solutions eventually led to further restrictions of completeness such that one did not really obtain a procedural counterpart to the declarative meaning of disjunction.

Within our framework, disjunction is most naturally and absolutely completely handled by the constraint solver itself. Our resolution algorithm does not at all know about the possible existence of disjunctive constraints. One standard example when discussing the treatment of disjunctive constraints in the literature is the minimum function. Its formulation as a CLP program is straightforward:

$$\min(X, Y, Z) \leftarrow (X \leqslant Y \wedge Z = X) \vee (Y \leqslant X \wedge Z = Y).$$

The answers that can be derived from this program are as complete and concise as the definition itself. For the query $\leftarrow \min(3, 4, Z)$ we obtain $Z - 3 = 0$. For $\leftarrow \min(X, Y, 3)$ the answer is

$$(X - 3 = 0 \wedge Y - 3 \geqslant 0) \vee (X - 3 \geqslant 0 \wedge Y - 3 = 0).$$

Asking for $\leftarrow \min(X, Y, Z)$, we obviously get the definition itself. These computations take no measurable time.

6.4 Quantified Constraints

Since our constraints are first-order formulas, they may also contain quantification. It follows, of course, from the existence of quantifier elimination procedures for our domains that this does not really increase expressivity. It, however, supports very much the concise formulation of programs.

The following program, for instance, describes that in real 2-space the point (U_1, U_2) is the image of the point (X_1, X_2) under central projection from the punctual light source (C_1, C_2):

$$\mathrm{pr}(C_1, C_2, X_1, X_2, U_1, U_2) \leftarrow \exists T\Big(T > 0 \land \bigwedge_{i=1}^{2} U_i = T(X_i - C_i)\Big).$$

Notice that this description covers all degenerate cases that arise when some of the points or coordinates happen to coincide. The following is a possible quantifier-free description with 10 atomic formulas:

$$\begin{aligned}&(C_1 = 0 \land C_2 = 0 \land U_1 = X_1 \land U_2 = X_2) \lor\\ &\quad(C_2 \neq 0 \land C_2U_2 > C_2X_2 \land C_1U_2 - C_1X_2 - C_2U_1 + C_2X_1 = 0) \lor\\ &\quad(C_1 \neq 0 \land C_1U_1 > C_1X_1 \land C_1U_2 - C_1X_2 - C_2U_1 + C_2X_1 = 0).\end{aligned}$$

The quantified formulation has been taken from [23], the quantifier-free result has been obtained by quantifier-elimination with REDLOG. In higher dimension the effect becomes more dramatic. The quantifier-free description in 3-space has 18 atomic formulas, the one in 4-space 28.

6.5 Beyond Real Numbers

We conclude our collection of examples with a non-real one. The following program over the domain of p-adic numbers is analogous to the definition of nat in Section 6.1.

$$\begin{aligned}\mathrm{ppow}(1) &\leftarrow\\ \mathrm{ppow}(p \cdot X) &\leftarrow \mathrm{ppow}(X),\ 1 \mid X.\end{aligned}$$

It defines the powers of the prime p. For each query, the constant p must be chosen to be a fixed prime; this completely determines the domain $\mathbb{Q}_p$. The constraint $1 \mid X$ states that X is a p-adic integer. It is obvious that successive division by p eventually leads to a number with negative value. Thus the constraint can play the role of the emergency brake such as $X \geqslant 0$ does in the definition of nat. For $p = 101$ the query

$$\leftarrow \mathrm{ppow}(12201900399479668244827490915525641902001)$$

yields “true” after 0.1 s. If we increase this number, which is 101^{20}, by 1, then the corresponding query immediately yields “false.” In this case, the constraint solver recognizes that $\frac{101^{20}+1}{101}$ is not a p-adic integer.

7 Conclusions

We have introduced an extension of CLP, where the constraints are arbitrary first-order formulas over some domain. Constraint solving is realized by various applications of quantifier elimination techniques. Our approach is implemented in our system CLP(RL). There are already various domains available there: $\mathbb{R}$, $\mathbb{Q}_p$, $\mathbb{C}$, $\mathbb{Z}$, quantified propositional calculus, and term algebras. The advantages of our approach include real of arbitrary degree, exact arithmetic, absolutely clean treatment of disjunction and other Boolean operators and, moreover, first-order quantified constraints. The idea behind our work is to consider the logic programming part as a supplement to quantifier elimination in order to provide solution methods for problems that have so far been accessible neither by quantifier elimination nor by constraint logic programming.

References

1. Dolzmann, A., Sturm, T., Weispfenning, V.: Real quantifier elimination in practice. In Matzat, B.H., Greuel, G.M., Hiss, G., eds.: Algorithmic Algebra and Number Theory. Springer, Berlin (1998) 221–247
2. Weispfenning, V.: Mixed real-integer linear quantifier elimination. In Dooley, S., ed.: Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (ISSAC 99), Vancouver, BC, ACM, ACM Press, New York, 1999 (1999) 129–136
3. Sturm, T., Weispfenning, V.: Quantifier elimination in term algebras. The case of finite languages. In Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V., eds.: Computer Algebra in Scientific Computing. Proceedings of the CASC 2002. TUM München (2002) 285–300
4. Hong, H.: RISC-CLP(Real): Constraint logic programming over real numbers. In Benhamou, F., Colmerauer, A., eds.: Constraint Logic Programming: Selected Research. MIT Press (1993)
5. Colmerauer, A. et al.: Workshop on first-order constraints, Marseille, France. Abstracts of talks (2001)
6. Dolzmann, A., Sturm, T.: Redlog: Computer algebra meets computer logic. ACM SIGSAM Bulletin **31** (1997) 2–9
7. Collins, G.E., Hong, H.: Partial cylindrical algebraic decomposition for quantifier elimination. Journal of Symbolic Computation **12** (1991) 299–328
8. Weispfenning, V.: The complexity of linear problems in fields. Journal of Symbolic Computation **5** (1988) 3–27
9. Weispfenning, V.: Quantifier elimination for real algebra—the quadratic case and beyond. Applicable Algebra in Engineering Communication and Computing **8** (1997) 85–101
10. Becker, E., Wörmann, T.: On the trace formula for quadratic forms. In Jacob, W.B., Lam, T.Y., Robson, R.O., eds.: Recent Advances in Real Algebraic Geometry and Quadratic Forms. Volume 155 of Contemporary Mathematics., American Mathematical Society, American Mathematical Society, Providence, Rhode Island (1994) 271–291 Proceedings of the RAGSQUAD Year, Berkeley, 1990–1991.

11. Pedersen, P., Roy, M.F., Szpirglas, A.: Counting real zeroes in the multivariate case. In Eysette, F., Galigo, A., eds.: Computational Algebraic Geometry. Volume 109 of Progress in Mathematics. Birkhäuser, Boston, Basel; Berlin (1993) 203–224 Proceedings of the MEGA 92.
12. Weispfenning, V.: A new approach to quantifier elimination for real algebra. In Caviness, B., Johnson, J., eds.: Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts and Monographs in Symbolic Computation. Springer, Wien, New York (1998) 376–392
13. Sturm, T.: Linear problems in valued fields. Journal of Symbolic Computation **30** (2000) 207–219
14. Weispfenning, V.: Comprehensive Gröbner bases. Journal of Symbolic Computation **14** (1992) 1–29
15. Weispfenning, V.: Complexity and uniformity of elimination in Presburger arithmetic. In Küchlin, W.W., ed.: Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (ISSAC 97), Maui, HI, ACM, ACM Press, New York, 1997 (1997) 48–53
16. Seidl, A., Sturm, T.: Boolean quantification in a first-order context. In Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V., eds.: Computer Algebra in Scientific Computing. Proceedings of the CASC 2003. TUM München (2003) 345–356
17. Dincbas, M., Van Hentenryck, P., Simonis, H., Aggoun, A., Graf, T., Berthier, F.: The constraint logic programming language CHIP. In: Proceedings of the International Conference on Fifth Generation Computer Systems, Tokyo, Japan, Decemeber 1988. Ohmsha Publishers, Tokyo (1988) 693–702
18. Jaffar, J., Michaylov, S., Stuckey, P.J., Yap, R.H.C.: The CLP(R) language and system. ACM Transactions on Programming Languages and Systems **14** (1992) 339–395
19. Colmerauer, A.: Prolog III. Communications of the ACM **33** (1990) 70–90
20. Dolzmann, A., Sturm, T.: Simplification of quantifier-free formulae over ordered fields. Journal of Symbolic Computation **24** (1997) 209–231
21. Frühwirth, T., Abdennadher, S.: Essentials of Constraint Programming. Springer (2003)
22. Kameny, S.L.: Roots: A reduce root finding package. In Hearn, A.C., Codemist Ltd., eds.: Reduce User's and Contributed Packages Manual Version 3.7. Anthony C. Hearn and Codemist Ltd. (1999) 513–518
23. Sturm, T., Weispfenning, V.: Computational geometry problems in Redlog. In Wang, D., ed.: Automated Deduction in Geometry. Volume 1360 of Lecture Notes in Artificial Intelligence (Subseries of LNCS). Springer-Verlag, Berlin Heidelberg (1998) 58–86

Computation of Full Comprehensive Gröbner Bases

Akira Suzuki

Kobe University
sakira@kobe-u.ac.jp

Abstract. In original comprehensive Gröbner bases, we must select parameter from variables before computation. By extending them, we introduce *full comprehensive Gröbner bases* as comprehensive Gröbner bases such that we can choose parameters to be instantiated from variables freely after computation. In this paper, we give an algorithm to compute full comprehensive Gröbner bases.

1 Introduction

In [10], V. Weispfenning introduced comprehensive Gröbner bases. A comprehensive Gröbner basis is a set of polynomials depending on parameters such that, for any instantiation of parameters, the set is a Gröbner basis of the instantiated system. Several algorithms to compute comprehensive Gröbner bases are introduced. [2,7] In general, a comprehensive Gröbner basis might not be a Gröbner basis. For example, if we let $F = \{A^2Y + 1, AX - Y\}$, then F is a comprehensive Gröbner basis in $\mathbb{C}[X, Y]$ with a parameter A, beside it is not a Gröbner basis in $\mathbb{C}[X, Y, A]$ since $X + Y^2A \in \langle F\rangle$. On the other hand, we can calculate a comprehensive Gröbner basis which is also a Gröbner basis as follow: Let K be a computable infinite field. Let $\bar{X} = \{X_1, \ldots, X_n\}$ be a set of variables, and $\bar{A} = \{A_1, \ldots, A_m\}$ be the set of parameters. Then, for a given finite set $F \subseteq K[\bar{X}]$, we can get a comprehensive Gröbner basis G of F in $K[\bar{X} \setminus \bar{A}]$ with parameters $\bar{A}$ such that $G \subseteq \langle F\rangle$ in $K[\bar{X}]$, and we can also get a Gröbner basis G' of the ideal $\langle F\rangle$ in $K[\bar{X}]$. Then we can easily see that $G \cup G'$ is a comprehensive Gröbner basis which is also a Gröbner basis.

For a convenience, we introduce terms as follow where $\mathcal{P}(\bar{X})$ is the power set of $\bar{X}$, i.e., $\mathcal{P}(\bar{X}) = \{\bar{A} \ : \ \bar{A} \subseteq \bar{X}\}$:

Definition 1. *Let $\bar{X}$ be a finite set of variables. Let K be a field. Let F and G be a finite subset of $K[\bar{X}]$. For $\bar{A} \subseteq \bar{X}$, G is $\bar{A}$-*comprehensive Gröbner basis *for F if G forms a comprehensive Gröbner basis for F with parameters $\bar{A}$. For $\mathcal{A} \subseteq \mathcal{P}(\bar{X})$, G is $\mathcal{A}$-*comprehensive Gröbner basis *for F if it is $\bar{A}$-comprehensive Gröbner basis for each $\bar{A} \in \mathcal{A}$. And G is a* full comprehensive Gröbner basis *for F if it is a $\mathcal{P}(\bar{X})$-comprehensive Gröbner basis for F.*

Full comprehensive Gröbner bases are introduced by the author [8]. Extending the argument of the previous paragraph, we can get a finite set $\bar{G} \subseteq K[\bar{X}]$

which is a full comprehensive Gröbner basis, by taking the union of the $\bar{B}$-comprehensive Gröbner bases $G_{\bar{B}}$ for each $\bar{B} \subseteq \bar{X}$. Then, if we consider any subset $\bar{B}$ of $\bar{X}$ as a parameters, $\bar{G}$ forms a $\bar{B}$-comprehensive Gröbner basis. Especially, considering the case $\bar{B} = \emptyset$, we can see that $\bar{G}$ is a Gröbner basis. For the $F \subseteq \mathbb{C}[X, Y, A]$ given in the beginning of this section, we get a set $\{X^2+Y^2, X+Y^2A, YA^2+1, XA-Y, XA^3+1\}$ gathering $2^3 = 8$ comprehensive Gröbner bases for F for various parameters. In this paper, we show a way to compute full comprehensive Gröbner bases which is more sophisticated than the naive method shown in this paragraph.

In this paper, we use definitions and arguments appearing in [7], so we assume the reader is familiar with them. We also assume the reader is familiar with a theory of Gröbner bases of polynomial rings over von Neumann regular rings [9,11,4] and a one of terraces and preterraces which are introduced in [6,5].

2 von Neumann Regular Rings

In this paper, we fix an algebraically closed field K and a finite set $\bar{X} = \{X_1, \ldots, X_n\}$ of variables. We also fix an admissible order $<_{\bar{X}}$ on the set of terms of $\bar{X}$. For a mapping $f\colon A \to B$ and $C \subseteq A$, we use the terminology $f[C]$ as the image of C under f, i.e., $f[C] = \{f(a) \in B \ : \ a \in C\}$.

A commutative ring R with identity 1 is *von Neumann regular ring* if R satisfies that $\forall a \in R \ \exists b \in R \quad a^2b = a$. For such a and b, defining $a^* = ab$ and $a^{-1} = ab^2$, we notice that a^* and a^{-1} are uniquely determined by a and independent from the choice of b. Note that every direct product of fields is a von Neumann regular ring. Then the set $K^{(K^n)}$ of the mappings from K^n into K forms a von Neumann regular ring.

A polynomial $f \in K[\bar{X}]$ can be considered as a mapping $f\colon K^n \to K$, i.e., a member of the ring $K^{(K^n)}$. Thus we can define the canonical embedding $\varphi_{\bar{X}}\colon K[\bar{X}] \to K^{(K^n)}$. Let $T_{\bar{X}}$ be the closure of the image $\varphi_{\bar{X}}[K[\bar{X}]]$ under addition, multiplication, and inverse of the von Neumann regular ring $(K^{(K^n)}, +, \cdot, {}^{-1})$. For $c \in T_{\bar{X}}$, we define the *support* of c by $\operatorname{supp}(c) = \{\bar{a} \in K^n \ : \ c(\bar{a}) \neq 0_K\}$.

For each subset $\bar{A}$ of $\bar{X}$, we let $\varphi_{\bar{A}}\colon K[\bar{A}] \to T_{\bar{X}}$ be the restriction of $\varphi_{\bar{X}}$ to $K[\bar{A}] \subseteq K[\bar{X}]$. Let $T_{\bar{A}} \subseteq T_{\bar{X}}$ be the closure of the image $\varphi_{\bar{A}}[K[\bar{A}]]$ under addition, multiplication, and inverse of the ring $(T_{\bar{X}}, +, \cdot, {}^{-1})$. Then we can easily see that $T_{\bar{A}}$ forms a von Neumann regular ring again. Let $T_{\bar{A}}^*$ be the set of idempotent elements of $T_{\bar{A}}$, i.e., $T_{\bar{A}}^* = \{c^* \ : \ c \in T_{\bar{A}}\}$. Let $T_{\bar{A}}^p$ be the set of the elements c of $T_{\bar{A}}^*$ which can be expressed by $c = \varphi_{\bar{X}}(f_1)^* \cdots \varphi_{\bar{X}}(f_k)^* \cdot (1 - \varphi_{\bar{X}}(g_1)^*) \cdots (1 - \varphi_{\bar{X}}(g_l)^*)$ for some $f_1, \ldots, f_k, g_1, \ldots, g_l \in K[\bar{A}]$. Then we have the following. We can show this from the fact the $\operatorname{supp}(c)$ is constructible.

Lemma 1. *Let $\bar{A} \subseteq \bar{X}$ and $c \in T_{\bar{A}}^*$. Then there are $d_1, \ldots, d_m \in T_{\bar{A}}^p$ such that $c = d_1 + \cdots + d_m$.*

Proof. First we note that c is a function of values 0 or 1. So it is enough to find $d_1, \ldots, d_m \in T_{\bar{A}}^p$ such that $\operatorname{supp}(c) = \operatorname{supp}(d_1) \sqcup \cdots \sqcup \operatorname{supp}(d_m)$ where $a = b \sqcup c$ means a is the disjoint union of b and c, i.e., $a = b \cup c$ and $b \cap c = \emptyset$.

Since $\mathrm{supp}(c)$ is algebraically constructible, it can be expressed by

$$\mathrm{supp}(c) = \bigcup_{i=1}^{N} (V(I_i) \setminus V(J_i)) \tag{1}$$

where $V(I_i)$ and $V(J_i)$ are varieties for ideals I_i and J_i on $K[\bar{A}]$ respectively. Then, by an easy argument, we may assume that (1) is a disjoint union. So it is enough to show that to find $d_1, \dots, d_m \in T^p_{\bar{A}}$ such that $V(I) \setminus V(J) = \mathrm{supp}(d_1) \sqcup \cdots \sqcup \mathrm{supp}(d_m)$ for any ideals I and J on $K[\bar{A}]$.

Say $I = \langle f_1, \dots, f_k \rangle$ and $J = \langle g_1, \dots, g_l \rangle$ where $f_1, \dots, f_k, g_1, \dots, g_l \in K[\bar{A}]$. If $l = 0$, then $V(I) \setminus V(J) = \emptyset = \varphi_{\bar{X}}(1)^*$, and so we are done. Thus we assume $l > 0$. Then $V(J) = V(g_1, \dots, g_l) = V(g_1) \cap \cdots \cap V(g_l)$ implies that

$$\begin{aligned} V(I) \setminus V(J) &= V(I) \setminus V(g_1, \dots, g_l) \\ &= V(I) \setminus (V(g_1) \cap \cdots \cap V(g_l)) \\ &= (V(I) \setminus V(g_1)) \cup \cdots \cup (V(I) \setminus V(g_l)) \\ &= (V(f_1, \dots, f_k) \setminus V(g_1)) \sqcup (V(f_1, \dots, f_k, g_1) \setminus V(g_2)) \sqcup \\ &\qquad\qquad V(f_1, \dots, f_k, g_1, \dots, g_{l-1}) \setminus V(g_l). \end{aligned}$$

So defining $d_1 = \varphi_{\bar{X}}(g_1)^* \cdot (1 - \varphi_{\bar{X}}(f_1)^*) \cdots (1 - \varphi_{\bar{X}}(f_k)^*)$, $d_2 = \varphi_{\bar{X}}(g_2)^* \cdot (1 - \varphi_{\bar{X}}(f_1)^*) \cdots (1 - \varphi_{\bar{X}}(f_k)^*) \cdot (1 - \varphi_{\bar{X}}(g_1)^*), \dots, d_l = \varphi_{\bar{X}}(g_l)^* \cdot (1 - \varphi_{\bar{X}}(f_1)^*) \cdots (1 - \varphi_{\bar{X}}(f_k)^* \cdot (1 - \varphi_{\bar{X}}(g_1)^*) \cdots (1 - \varphi_{\bar{X}}(g_{l-1})^*)$, we have that $V(I) \setminus V(J) = \mathrm{supp}(d_1) \sqcup \cdots \sqcup \mathrm{supp}(d_l)$ and that $d_1, \dots, d_l \in T^p_{\bar{A}}$. □

When we assume that $\bar{A} \subseteq \bar{X}$, we can express $\bar{A} = \{X_{i_1}, X_{i_2}, \dots, X_{i_m}\} \subseteq \{X_1, X_2, \dots, X_n\} = \bar{X}$ with $i_1 < i_2 < \cdots < i_m$. Then $\mathrm{proj}_{\bar{A}} \colon K^n \to K^m$ is defined by $\mathrm{proj}((a_1, \dots, a_n)) = (a_{i_1}, \dots, a_{i_m})$. We also define, for $s \subseteq K^n$, $\mathrm{proj}_{\bar{A}}(s) = \{\mathrm{proj}_{\bar{A}}(\bar{a}) \in K^m \ : \ a \in s\}$. Furthermore, we define a mapping $\pi_{\bar{A}} \colon T^*_{\bar{X}} \to T^*_{\bar{A}}$ such that $\mathrm{supp}(\pi_{\bar{A}}(c)) = \bigcap \{\mathrm{supp}(a) \subseteq K^n \ : \ a \in T^*_{\bar{A}}, \mathrm{supp}(c) \subseteq \mathrm{supp}(a)\}$. Then we can see the well-definedness of it by the argument in [6] or [5].

We fix an arbitrary subset $\bar{A}$ of $\bar{X}$ and a polynomial f in $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$. We define the term order $<_{\bar{X} \setminus \bar{A}}$ on the set of terms of $\bar{X} \setminus \bar{A}$ by the restriction of $<_{\bar{X}}$. Then the *leading term* of f is a term of $\bar{X} \setminus \bar{A}$ is denoted by $lt(f)$. The *leading coefficient* $lc(f)$ is a member of $T_{\bar{X}}$, and the *leading monomial* $lm(f)$ satisfies $lm(f) = lc(f) \cdot lt(f)$. We often use Greek letters α, β, γ for terms of $\bar{X} \setminus \bar{A}$, alphabet letters f, g, h for polynomials in $K[\bar{X}]$, and a, b, c for elements of $T^*_{\bar{X}}$.

3 Comprehensive Pairs and Gröbner Bases

In this section, we use the symbol $\bar{A}$ for a subset of $\bar{X}$. Comprehensive pairs are a generalization of "witnessing pairs of coherent polynomials" appeared in [7]. The reader refers it for detailed arguments. In this paper, each pair (c, f) in

$T^p_{\bar{X}} \times K[\bar{X}]$ is called a *comprehensive pair.* For $(c, f) \in T^*_{\bar{X}} \times K[\bar{X}]$, we define $\psi_{\bar{A}}((c, f)) = c \cdot \varphi_{\bar{A}}(f) \in T_{\bar{X}}[\bar{X} \setminus \bar{A}]$ and $\tilde{\psi}_{\bar{A}}((c, f)) = \pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(f) \in T_{\bar{A}}[\bar{X} \setminus \bar{A}]$.

From now on, we give a sketch of an algorithm to compute a finite subset G of $T^p_{\bar{X}} \times K[\bar{X}]$ from a given finite set $F \subseteq K[\bar{X}]$ and $\mathcal{A} \subseteq \mathcal{P}(\bar{X})$ ($\mathcal{P}(\bar{X})$ is the power set of $\bar{X}$, the set of the all subsets of $\bar{X}$) such that:

1. For each $\bar{A} \in \mathcal{A}$, the set $\tilde{\psi}_{\bar{A}}[G]$ forms a Gröbner basis of $\langle \varphi_{\bar{A}}[F] \rangle$, the ideal generated by the image of F under $\varphi_{\bar{A}}$, in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$,
2. $\{g \;:\; \exists c \; (c, g) \in G\} \subseteq \langle F \rangle$ in $K[\bar{X}]$.

We see that the set G above gives us $\mathcal{A}$-comprehensive Gröbner Basis for F in the latter section.

Definition 2. *Let P be a finite subset of $T^p_{\bar{X}} \times K[\bar{X}]$. Let F be a finite set of polynomials in $K[\bar{X}]$. Then P is* $\bar{A}$-faithful *to F if*

1. $\langle \psi_{\bar{A}}[P] \rangle = \langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$,
2. $f \in \langle F \rangle$ for each $(c, f) \in P$.

We say P is $\mathcal{A}$-faithful *to F if P is $\bar{A}$-faithful to F for each $\bar{A} \in \mathcal{A}$.*

The following Lemma is used to check faithfulness of sets of comprehensive pairs.

Lemma 2. *Let $\bar{A} \subseteq \bar{B} \subseteq \bar{X}$. Let F be a finite subset of $K[\bar{X}]$. Let P be a finite subset of $T^*_{\bar{B}} \times K[\bar{X}]$. If $\{g \;:\; \exists c \; (c, g) \in P\} \subseteq \langle F \rangle$ in $K[\bar{X}]$, then $\psi_{\bar{A}}[P] \subseteq \langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{B}}[\bar{X} \setminus \bar{A}]$.*

Proof. We pick arbitrary $(c, g) \in P$. Then we know that $g \in \langle F \rangle$ in $K[\bar{X}]$ by the assumption. So we can take polynomials $f_1, \ldots, f_m \in F$ and $h_1, \ldots, h_m \in K[\bar{X}]$ such that $g = f_1 h_1 + \cdots f_m h_m$. Since $\varphi_{\bar{A}} \colon (K[\bar{A}])[\bar{X} \setminus \bar{A}] \to T_{\bar{A}}[\bar{X} \setminus \bar{A}]$ is an embedding, we have $\varphi_{\bar{A}}(g) = \varphi_{\bar{A}}(f_1)\varphi_{\bar{A}}(h_1) + \cdots + \varphi_{\bar{A}}(f_m)\varphi_{\bar{A}}(h_m)$. So we have $\varphi_{\bar{A}}(g) \in \langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$. Thus we have $\psi_{\bar{A}}((c, g)) = c \cdot \varphi_{\bar{A}}(g) \in \langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{B}}[\bar{X} \setminus \bar{A}]$ since $c \in T^*_{\bar{B}} \supseteq T^*_{\bar{A}}$. □

We can easily see that $\{1_{T_{\bar{X}}}\} \times F$ is $\mathcal{P}(\bar{X})$-faithful to F for any finite $F \subseteq K[\bar{X}]$. So we give a method to calculate a finite set G of comprehensive pairs appeared in the beginning of this section by modifying Buchberger algorithm, starting from $\{1\} \times F$ keeping the comprehensive pairs to be faithful to F.

Definition 3. *Let* op *be a mapping from a finite set of comprehensive pairs to a finite set of comprehensive pairs (or a comprehensive pair). We say* $\mathrm{op}((c_1, f_1), \ldots, (c_m, f_m)) = \{(d_1, g_1), \ldots, (d_l, g_l)\}$. *Then* op *is* faithful *if $g_1, \ldots, g_l \in \langle f_1, \cdots, f_m \rangle$.*

In the rest of this section, we show faithful operations to calculate a finite set G of comprehensive pairs such that $\psi_{\bar{A}}[G]$ forms a Gröbner basis in $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$. We should note that we does not require $\psi_{\bar{A}}[G]$ to be a Gröbner basis in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$ at the present moment.

Definition 4. *Let* (c, f) *and* (d, g) *be comprehensive pairs. We define an operation* Mul *by* $\mathrm{Mul}((c, f), (d, g)) = (cd, fg)$*.*

Definition 5. *Let* (c, f) *and* (c, g) *be comprehensive pairs which share their first coordinate. We define an operation* Add *by* $\mathrm{Add}((c, f), (c, g)) = (c, f + g)$*.*

Then we can easily see the following Proposition for these operations.

Proposition 1. *Let* $c, d \in T^p_{\bar{X}}$ *and* $f, g \in K[\bar{X}]$*. Then, for each* $\bar{A} \subseteq \bar{X}$*, we have*

1. $\psi_{\bar{A}}((c, f)) \cdot \psi_{\bar{A}}((d, g)) = \psi_{\bar{A}}(\mathrm{Mul}((c, f), (d, g))$,
2. $\psi_{\bar{A}}((c, f)) + \psi_{\bar{A}}((c, g)) = \psi_{\bar{A}}(\mathrm{Add}((c, f), (c, g))$,

in $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$*. We also have that* Mul *and* Add *are faithful.*

Next we consider boolean closures for comprehensive pairs. A polynomial $f \in T_{\bar{X}}[\bar{X}]$ is *boolean closed* if $f = lc(f)^* \cdot f$, and *boolean closure* of f is $bc(f) = lc(f)^* \cdot f$. For a finite set F of polynomials in $T_{\bar{X}}[\bar{X}]$, a finite set $bc(F)$ of polynomials in $T_{\bar{X}}[\bar{X}]$ is called *boolean closure* of F if $\langle F \rangle = \langle bc(F) \rangle$ in $T_{\bar{X}}[\bar{X}]$ and if each polynomial in $bc(F)$ is boolean closed, though $bc(F)$ is not uniquely determined by F. As you can see in the following theorem, boolean closure is an important operation to calculate Gröbner bases in polynomial rings over von Neumann regular rings.

Theorem 1. *Let* G *be a finite set of boolean-closed polynomials. Then* G *is a Gröbner basis iff* $SPol(f, g) \xrightarrow{*}_G 0$ *for any pair* f *and* g *in* G*.*

We first see an algorithm to compute a boolean closure $bc(\{\psi_{\bar{A}}(c, f)\})$ of the singleton of a comprehensive pair with respect to $\bar{A}$. We can easily see that the operator defined by this **BCSingle** is faithful since Mul is so.

```
                    Algorithm. BCSingle
Input: $c \in T^p_{\bar{X}}$, $f \in K[\bar{X}]$, and $\bar{A} \subseteq \bar{X}$
Output: $Q$ : a finite set of comprehensive pairs of
    boolean-closed polynomials with respect to $\bar{A}$ such that
    $\langle \psi_{\bar{A}}(c, f) \rangle = \langle \psi_{\bar{A}}[Q] \rangle$

If $\psi_{\bar{A}}(c, f) = 0$ then
  Return $\emptyset$;
Else
  Return $\{\mathrm{Mul}((c, f), (lc(\psi_{\bar{A}}(c, f))^*, 1)\} \cup$
    BCSingle$(\mathrm{Mul}((c, f), (1 - lc(\psi_{\bar{A}}(c, f))^*, 1), \bar{A})$;
```

For a finite set P of comprehensive pairs and $\bar{A} \subseteq \bar{X}$, we define the boolean closure of it with respect to $\bar{A}$ by $bc_{\bar{A}}(P) = \bigcup_{p \in P} \mathbf{BCSingle}(p, \bar{A})$. For $\bar{A} \subseteq \bar{X}$, a comprehensive pair p is $\bar{A}$-*boolean closed* if $\psi_{\bar{A}}(p)$ is boolean closed in $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$. And, for $\mathcal{A} \subseteq \mathcal{P}(\bar{X})$, a comprehensive pair p is $\mathcal{A}$-*boolean closed* if p is $\bar{A}$-boolean closed for each $\bar{A} \in \mathcal{A}$. For a boolean closed polynomial $f \in T_{\bar{X}}[\bar{X}]$ and $c \in T^*_{\bar{X}}$, we can easily see that $c \cdot f$ is boolean closed. On the other hand, the boolean closure above only require multiplications by elements of $T^p_{\bar{X}}$ essentially. From these facts, we can get an algorithm **FullBC** such that $\langle \psi_{\bar{A}}[P] \rangle = \langle \psi_{\bar{A}}[Q] \rangle$ for each $\bar{A} \in \mathcal{A}$, where P is a finite set of comprehensive pairs, $Q = \mathbf{FullBC}(P)$, and $\mathcal{A} \subseteq \mathcal{P}(\bar{X})$.

The computations of the S-polynomials and the normal form is essentially same as the ones shown in [7], though we have to distinct the ones with respect to $\bar{A}$ for each $\bar{A} \subseteq \bar{X}$, e.g., $\mathrm{SPol}_{\bar{A}}(f, g) \neq \mathrm{SPol}_{\bar{B}}(f, g)$ may occur. For monomial reductions of comprehensive pairs, we consider a polynomial $f = b\alpha\beta + f'$ and a boolean-closed polynomial $g = a\alpha + g'$ where $a \cdot b \neq 0$ and $lm(g) = a\alpha$. So we first split f to $a^* f$ and $(1 - a^*)f$ (*split stage*), then we concentrate to apply monomial reduction to af by bg as $af \longrightarrow_{bg} af - b\beta g = af' - b\beta g'$. (*reduction stage*) Moreover, we can assume that, for any $\bar{A} \subseteq \bar{X}$, it outputs the empty set if the $0_{T_{\bar{X}}[\bar{X} \setminus \bar{A}]}$ is given as its input, i.e., $\mathbf{NormalForm}_{\bar{A}}((c, f), P) = \emptyset$ for each $\bar{A} \subseteq \bar{X}$ and $(c, f) \in T^p_{\bar{X}} \times K[\bar{X}]$ with $c \cdot \varphi_{\bar{A}}(f) = 0_{T_{\bar{X}}[\bar{X} \setminus \bar{A}]}$ and for any finite $P \subseteq T^p_{\bar{X}} \times K[\bar{X}]$.

Then we can give the faithful algorithm **GröbnerBasis** as below:

Algorithm. GröbnerBasis

```
Input: $F$ : a finite subset of $K[\bar{X}]$, $\mathcal{A} \subseteq \mathcal{P}(\bar{X})$
Output: $G$ : a finite set of comprehensive pairs such that
    $\psi_{\bar{A}}[G]$ forms a Gröbner basis of $\langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$ for each $\bar{A} \in \mathcal{A}$
    and that $G$ is faithful to $F$

$G := \mathbf{FullBC}(\{1_{T_{\bar{X}}}\} \times F, \mathcal{A})$;
$G' := \emptyset$;
While $G \neq G'$ do
  $G' := G$;
  For each $\bar{A} \in \mathcal{A}$ do
    For each $p, q \in G$ with $p \neq q$ do
      $N := \mathbf{NormalForm}_{\bar{A}}(\mathbf{SPol}_{\bar{A}}(p, q), G)$;
      $G := G \cup \mathbf{FullBC}(N, \mathcal{A})$;
    End
  End
End
Return $G$;
```

In order to see that the output G of this algorithm is a one as in the requirement of "**Output:**" part, we first show that G is faithful to F during the computation. Since **NormalForm** and **FullBC** are faithful operations, we have that $\{g \ : \ \exists c \ (c, g) \in G\} \subseteq F$. So it is enough to show that $\langle \psi_{\bar{A}}[G] \rangle = \langle \varphi_{\bar{A}}[F] \rangle$ for each $\bar{A} \in \mathcal{A}$. So we fix $\bar{A} \in \mathcal{A}$. From the fact $\mathbf{FullBC}(\{1\} \times F, \mathcal{A}) \subseteq G$, we see that $\langle \psi_{\bar{A}}[G] \rangle \supseteq \langle \varphi_{\bar{A}}[F] \rangle$. On the other hand, we can see $\langle \psi_{\bar{A}}[G] \rangle \subseteq \langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$ by applying Lemma 2 with $\bar{B} = \bar{X}$.

Then, for each fixed $\bar{A} \in \mathcal{A}$, in the While-loop in this algorithm, though new comprehensive pairs may be added to G as normal forms of S-polynomials, the other comprehensive pairs may also be added to G with respect to $\bar{B} \in \mathcal{A} \setminus \{\bar{A}\}$. We can see that the extra comprehensive pairs for $\bar{A}$ does not affect to $\psi_{\bar{A}}[G]$ to be a Gröbner basis for $\psi_{\bar{A}}[F]$, since G is faithful to F and so $\psi_{\bar{A}}[G] \subseteq \langle \varphi_{\bar{A}}[F] \rangle$ holds during the computation. So, for each $\bar{A} \in \mathcal{A}$, the procedure to add new comprehensive pairs stops in finite steps by the same reason of one of Buchberger's Algorithm. Thus we can see that this algorithm **GröbnerBasis** stops in finite steps since $|\mathcal{A}| \leq 2^n$ is finite, and that $\psi_{\bar{A}}[G]$ forms a Gröbner basis for $\langle \varphi_{\bar{A}}[F] \rangle$ in the polynomial ring $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$ (not in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$) for each $\bar{A} \in \mathcal{A}$.

4 Comprehensive Gröbner Bases

In this section, we first show, for a given finite $F \subseteq K[\bar{X}]$ and $\mathcal{A} \subseteq \mathcal{P}(\bar{X})$, that the algorithm **GröbnerBasis** outputs G required in the beginning of the last section. Note that we have seen, at the end of the last section, that the G satisfies

1. $\psi_{\bar{A}}[G]$ forms a Gröbner basis, and so $\tilde{\psi}_{\bar{A}}[G]$ forms a Gröbner basis of $\langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$ for each $\bar{A} \in \mathcal{A}$, and
2. $\{g \,:\, \exists c\, (c, g) \in G\} \subseteq \langle F \rangle$ in $K[\bar{X}]$.

Therefore it is enough to show that $\tilde{\psi}_{\bar{A}}[G]$ is a Gröbner basis for $\langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$, not in $T_{\bar{X}}[\bar{X} \setminus \bar{A}]$, for each $\bar{A} \in \mathcal{A}$. To show it, we introduce a concept of $\bar{A}$-principal and show some Lemmas concerning it.

Definition 6. *Let $\bar{A} \subseteq \bar{X}$. Then $c \in T^*_{\bar{X}}$ is* $\bar{A}$-principal *if $c \cdot \varphi_{\bar{X}}(h) \neq 0$ holds for any $h \in K[\bar{X}] \setminus K[\bar{A}]$.*

Lemma 3. *Let $\bar{A} \subseteq \bar{X}$. Let $c \in T^p_{\bar{X}}$ be an $\bar{A}$-principal element such that $\pi_{\bar{A}}(c) \in T^p_{\bar{A}}$. Then there is a polynomial $f \in (K[\bar{X}] \setminus K[\bar{A}]) \cup \{1_{K[\bar{X}]}\}$ such that $c = \pi_{\bar{A}}(c) \cdot \varphi_{\bar{X}}(f)^*$ in $T_{\bar{X}}$.*

Proof. In this proof, we abbreviate $\varphi_{\bar{X}}$ to φ for the readability if there is no confusion.

From the assumption $c \in T^p_{\bar{X}}$, we get $f'_1, \ldots, f'_{k'}, g'_1, \ldots, g'_{l'} \in K[\bar{A}]$ such that $c = \varphi(f'_1)^* \cdots \varphi(f'_{k'})^* \cdot (1 - \varphi(g'_1)^*) \cdots (1 - \varphi(g'_{l'})^*)$. Since $\mathrm{supp}(c) \subseteq \mathrm{supp}(\pi_{\bar{A}}(c))$ from the definition of $\pi_{\bar{A}}$, we have $c = \pi_{\bar{A}}(c) \cdot c$. So we have

$$c = \pi_{\bar{A}}(c) \cdot \varphi(f'_1)^* \cdots \varphi(f'_{k'})^* \cdot (1 - \varphi(g'_1)^*) \cdots (1 - \varphi(g'_{l'})^*.$$

We let $f = f'_1 \cdots f'_{k'} \in K[\bar{X}]$. Then we notice that $\varphi(f)^* = \varphi(f'_1 \cdots f'_{k'})^* = \varphi(f'_1)^* \cdots \varphi(f'_{k'})^*$ by an easy argument. So we have the following:

$$c = \pi_{\bar{A}}(c) \cdot \varphi(f) \cdot (1 - \varphi(g'_1)^*) \cdots (1 - \varphi(g'_{l'})^*). \tag{2}$$

Then we notice the following.

Claim. We may assume $g'_1, \ldots, g'_{l'} \notin K[\bar{A}]$.

Proof (Claim). We suppose $g'_i \in K[\bar{A}]$ for some $i = 1, \ldots, l'$. Then we know that $\pi_{\bar{A}}(c) \cdot (1 - \varphi(g'_i)^*) \in T^p_{\bar{A}}$ and that $\operatorname{supp}(c) \subseteq \operatorname{supp}(\pi_{\bar{A}}(c) \cdot (1 - \varphi(g'_i)^*))$. Thus we have $\pi_{\bar{A}}(c) \cdot (1 - \varphi(g'_i)^*) = \pi_{\bar{A}}(c)$ by the definition of $\pi_{\bar{A}}$. So if $g'_i \in K[\bar{A}]$, then it can be omitted from (2). $\dashv$*(Claim)*

Thus we assume $g'_1, \ldots, g'_{l'} \in K[\bar{X}] \setminus K[\bar{A}]$. Then we have the following.

Claim. We have $l' = 0$, i.e., there is no g'_i in (2).

Proof (Claim). We suppose that $g'_i \in K[\bar{X}] \setminus K[\bar{A}]$ exists satisfying (2) for a contradiction. Then we knew that $\varphi(g'_i) \cdot (1 - \varphi(g'_i)^*) = \varphi(g'_i) - \varphi(g'_i) = 0$ in $T_{\bar{X}}$. So we had $c \cdot \varphi(g'_i) = 0$, which contradicts to the assumption that c is $\bar{A}$-principal. $\dashv$*(Claim)*

So c can be expressed by $c = \pi_{\bar{A}}(c) \cdot \varphi(f)^*$. If $f \in K[\bar{A}]$, then $\operatorname{supp}(c) \subseteq \operatorname{supp}(\pi_{\bar{A}}(c) \cdot \varphi(f))$ and so $\pi_{\bar{A}}(c) \cdot \varphi(f)^* = \pi_{\bar{A}}(c)$ by the definition of $\pi_{\bar{A}}$, and so we may assume that $f = 1_{K[\bar{X}]}$. Therefore we may assume that $f \in (K[\bar{X}] \setminus K[\bar{A}]) \cup \{1_{K[\bar{X}]}\}$. □

Lemma 4. *Let $\bar{A} \subseteq \bar{X}$. Let $c, d \in T^p_{\bar{X}}$ be $\bar{A}$-principal elements such that $\pi_{\bar{A}}(c) \cdot \pi_{\bar{A}}(d) \neq 0$. Then $c \cdot d \in T^p_{\bar{X}}$ is $\bar{A}$-principal and $\pi_{\bar{A}}(c \cdot d) = \pi_{\bar{A}}(c) \cdot \pi_{\bar{A}}(d)$.*

Proof. In this proof, we abbreviate $\varphi_{\bar{X}}$ to φ. Since $\pi_{\bar{A}}(c) \cdot \pi_{\bar{A}}(d) \in T^*_{\bar{A}}$, we have $p_1, \ldots, p_l \in T^p_{\bar{A}}$ such that $\pi_{\bar{A}}(c) \cdot \pi_{\bar{A}}(d) = p_1 + \cdots + p_l$ in $T_{\bar{A}}$ by Lemma 1. Then, for each $i = 1, \ldots, l$, we notice that $p_i \cdot c \in T^p_{\bar{X}}$ and that $\pi_{\bar{A}}(p_i \cdot c) = p_i \in T^p_{\bar{A}}$ since $\operatorname{supp}(p_i) \subseteq \operatorname{supp}(\pi_{\bar{A}}(c))$. So we can take $f_i \in (K[\bar{X}] \setminus K[\bar{A}]) \cup \{1_{K[\bar{X}]}\}$ such that $p_i \cdot c = p_i \cdot \varphi(f_i)^*$ by Lemma 3, for each $i = 1, \ldots, l$. In the same way, we can take $g_1, \ldots, g_l \in (K[\bar{X}] \setminus K[\bar{A}]) \cup \{1_{K[\bar{X}]}\}$ such that $p_i \cdot d = p_i \cdot \varphi(g_i)^*$ for each $i = 1, \ldots, l$.

Fixing i with $1 \leq i \leq l$ arbitrarily, we show that $(p_i \cdot c) \cdot (p_i \cdot d)$ is $\bar{A}$-principal and that $\pi_{\bar{A}}((p_i \cdot c) \cdot (p_i \cdot d)) = p_i$ since $\pi_{\bar{A}}(p_i \cdot c) = \pi_{\bar{A}}(p_i \cdot d) = p_i$. Now we have $(p_i \cdot c) \cdot (p_i \cdot d) = p_i \cdot \varphi(f_i)^* \cdot p_i \cdot \varphi(g_i)^* = p_i \cdot \varphi(f_i \cdot g_i)^*$. So we can see that it is $\bar{A}$-principal. If $f_i \cdot g_i = 1_{K[\bar{X}]}$, then we have nothing to do, so we assume $f_i \cdot g_i \in K[\bar{X}] \setminus K[\bar{A}]$. Since it is trivial that $\operatorname{supp}(p_i \cdot \varphi(f_i \cdot g_i)^*) \subseteq \operatorname{supp}(p_i)$, it is enough to show that $\operatorname{proj}_{\bar{A}}(\operatorname{supp}(p_i)) \subseteq \operatorname{proj}_{\bar{A}}(\operatorname{supp}(\pi_{\bar{A}}(p_i \cdot \varphi(f_i \cdot g_i)^*)))$. So we fix an arbitrary $\bar{a} \in \operatorname{proj}_{\bar{A}}(\operatorname{supp}(p_i))$. Then the instantiated polynomial $(f_i \cdot g_i)(\bar{a}, \bar{X} \setminus \bar{A})$ is a member of $K[\bar{X} \setminus \bar{A}] \setminus K$, and so $\{\bar{b} \in K^{|\bar{X} \setminus \bar{A}|} : (f_i \cdot g_i)(\bar{a}, \bar{b}) \neq 0\} \neq \emptyset$. It means that $\bar{a} \in \operatorname{proj}_{\bar{A}}(\operatorname{supp}(\pi_{\bar{A}}(\varphi(f_i \cdot g_i)^*)))$. So we have $\bar{a} \in \operatorname{proj}_{\bar{A}}(\operatorname{supp}(\pi_{\bar{A}}(p_i \cdot \varphi(f_i \cdot g_i)^*)))$.

Then, for any $h \in K[\bar{X}] \setminus K[\bar{A}]$, we have $p_1 \cdot \varphi(f_1 \cdot g_1)^* \cdot \varphi(h)^* \neq 0$ by the previous paragraph, and so $(c \cdot d) \cdot \varphi(h)^* \neq 0$, i.e., $c \cdot d$ is $\bar{A}$-principal. We also know that $c \cdot d = p_1 \cdot \varphi(f_1 \cdot g_1)^* + \cdots + p_l \cdot \varphi(f_l \cdot g_l)^*$ by the previous paragraph and the fact $p_i \cdot p_j = 0$ if $i \neq j$. Therefore we have $\pi_{\bar{A}}(c \cdot d) = \pi_{\bar{A}}(p_1 \cdot \varphi(f_1 \cdot g_1)^*) + \cdots + \pi_{\bar{A}}(p_l \cdot \varphi(f_l \cdot g_l)^*) = p_1 + \cdots p_l = \pi_{\bar{A}}(c) \cdot \pi_{\bar{A}}(d)$. □

Lemma 5. *Let $a \in T^p_{\bar{X}}$ and $b, c \in T^*_{\bar{X}}$ be such that a is $\bar{A}$-principal, c is not $\bar{A}$-principal, and that $a = b + c$ in $T_{\bar{X}}$. Then b is $\bar{A}$-principal and $\pi_{\bar{A}}(a) = \pi_{\bar{A}}(b)$.*

Proof. Using Lemma 1 for $\pi_{\bar{A}}(a) \in T^*_{\bar{A}}$, we can get $p_1, \ldots, p_l \in T^p_{\bar{A}}$ such that $\pi_{\bar{A}}(a) = p_1 + \cdots + p_l$. Then since $a \cdot p_i$ is $\bar{A}$-principal and $\pi_{\bar{A}}(a \cdot p_i) = p_i$ by Lemma 4, and so we can take $f_i \in (K[\bar{X}] \setminus K[\bar{A}]) \cup \{1_{K[\bar{X}]}\}$ such that $p_i \cdot a = p_i \cdot \varphi_{\bar{X}}(f_i)^*$ by Lemma 3, for each $i = 1, \ldots, l$. On the other hand, since c is not $\bar{A}$-principal, we can take $g \in K[\bar{X}] \setminus K[\bar{A}]$ such that $c \cdot \varphi_{\bar{X}}(g)^* = 0$. Then $\mathrm{supp}(c) \subseteq \mathrm{supp}(a \cdot (1 - \varphi_{\bar{X}}(g)^*))$. Therefore, for each $i = 1, \ldots, l$,

$$\begin{aligned}\mathrm{supp}(b \cdot p_i) &= \mathrm{supp}(a \cdot p_i) \setminus \mathrm{supp}(c)\\ &\supseteq \mathrm{supp}(a \cdot p_i \cdot \varphi_{\bar{X}}(g)^*)\\ &= \mathrm{supp}(p_i \cdot \varphi_{\bar{X}}(f_i \cdot g)^*).\end{aligned}$$

Therefore we can see that $b \cdot p_i$ is $\bar{A}$-principal and that $\pi_{\bar{A}}(b \cdot p_i) = p_i = \pi_{\bar{A}}(a \cdot p_i)$ since $f_i \cdot g \in K[\bar{X}] \setminus K[\bar{A}]$ for each $i = 1, \ldots, l$. So b is $\bar{A}$-principal and $\pi_{\bar{A}}(a) = \pi_{\bar{A}}(b)$. □

Lemma 6. *Let $c, d \in T^p_{\bar{X}}$ be $\bar{A}$-principal elements such that $\pi_{\bar{A}}(c) = \pi_{\bar{A}}(d)$. For $f, g \in K[\bar{X}]$, if $c \cdot \varphi_{\bar{A}}(f) = d \cdot \varphi_{\bar{A}}(g)$, then $\pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(f) = \pi_{\bar{A}}(d) \cdot \varphi_{\bar{A}}(g)$ in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$.*

Proof. We assume that $c \cdot \varphi_{\bar{A}}(f) = d \cdot \varphi_{\bar{A}}(g)$. We take $p_1, \ldots, p_l \in T^p_{\bar{A}}$ such that $\pi_{\bar{A}}(c) = p_1 + \cdots + p_l$ by Lemma 1. Since c and d are $\bar{A}$-principal, we can take $f_1, \ldots, f_l, g_1, \ldots, g_l \in (K[\bar{X}] \setminus K[\bar{A}]) \cup \{1_{K[\bar{X}]}\}$ by Lemma 3 such that $c \cdot p_i = p_i \cdot \varphi_{\bar{X}}(f_i)^*$ and that $d \cdot p_i = p_i \cdot \varphi_{\bar{X}}(g_i)^*$ for each $i = 1, \ldots, l$.

We fix i with $1 \leq i \leq l$ arbitrarily. Let $\bar{a} \in \mathrm{proj}_{\bar{A}}(p_i)$ arbitrarily. Since $(c \cdot d) \cdot \varphi_{\bar{A}}(f) = (c \cdot d) \cdot \varphi_{\bar{A}}(g)$, we have

$$\forall b \in K^{|\bar{X} \setminus \bar{A}|} \quad \big((f_i(\bar{a}, \bar{b}) \neq 0 \wedge g_i(\bar{a}, \bar{b}) \neq 0) \longrightarrow f(\bar{a}, \bar{b}) = g(\bar{a}, \bar{b})\big).$$

This is equivalent to

$$\forall b \in K^{|\bar{X} \setminus \bar{A}|} \quad \big(f_i(\bar{a}, \bar{b}) = 0 \vee g_i(\bar{a}, \bar{b}) = 0 \vee (f - g)(\bar{a}, \bar{b}) = 0\big).$$

Since K is an infinite field and $f_i(\bar{a}, \bar{X} \setminus \bar{A}), g_i(\bar{a}, \bar{X} \setminus \bar{A}) \in (K[\bar{X} \setminus \bar{A}] \setminus K) \cup \{1\}$, we have $(f - g)(\bar{a}, \bar{X} \setminus \bar{A}) = 0_{K[\bar{X} \setminus \bar{A}]}$. Since the choice of $\bar{a} \in \mathrm{proj}_{\bar{A}}(p_i)$ is arbitrary, we have $p_i \cdot \varphi_{\bar{A}}(f) = p_i \cdot \varphi_{\bar{A}}(g)$.

Since the choice of i is arbitrary, we have $\pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(f) = \pi_{\bar{A}}(d) \cdot \varphi_{\bar{A}}(g)$. □

Then we have the following:

Theorem 2. *Let F be a finite subset of $K[\bar{X}]$ and $\mathcal{A}$ be a subset of $\mathcal{P}(\bar{X})$. Let G be the finite set of comprehensive pairs such that $G =$ **GröbnerBasis**$(F, \mathcal{A})$. Then, for each $\bar{A} \in \mathcal{A}$, we have that $\tilde{\psi}_{\bar{A}}[G]$ forms a Gröbner basis of the ideal $\langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$. And so $\tilde{\psi}_{\bar{A}}[G]$ is an ACGB for F with respect to parameters $\bar{A} \in \mathcal{A}$.*

Proof. We fix $\bar{A} \in \mathcal{A}$ arbitrarily and show that $\tilde{\psi}_{\bar{A}}[G]$ is a Gröbner basis of $\langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$.

We first note that the fact $\{(\pi_{\bar{A}}(c), g) : (c, g) \in G\} \subseteq T^*_{\bar{A}} \times K[\bar{X}]$ implies that $\tilde{\psi}_{\bar{A}}[G] = \psi_{\bar{A}}[\{(\pi_{\bar{A}}(c), g) : (c, g) \in G\}] \subseteq \langle \varphi_{\bar{A}}[F] \rangle$ in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$ by applying Lemma 2 with $\bar{B} = \bar{A}$. Then we have the following:

Claim. For each $(c, f) \in G$, we have $\pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(f)$ is boolean closed in $T^*_{\bar{A}}[\bar{X} \setminus \bar{A}]$.

Proof (Claim). Let $g \in K[\bar{A}]$ be such that $lc(c \cdot \varphi_{\bar{A}}(f)) = c \cdot \varphi_{\bar{X}}(g)$ and that g divides f. By the definition of **BCSingle**, we have $c \cdot \varphi_{\bar{X}}(g)^* = c$. Then $\pi_{\bar{A}}(c) = \pi_{\bar{A}}(c \cdot \varphi_{\bar{X}}(g)^*) = \pi_{\bar{A}}(c) \cdot \varphi_{\bar{X}}(g)^*$ by the definition of $\pi_{\bar{A}}$. On the other hand, since $lc(\pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(f)) = \pi_{\bar{A}}(c) \cdot \varphi_{\bar{X}}(g)$, we have that $lc(\pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(f))^* \cdot (\pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(f)) = \pi_{\bar{A}}(c) \cdot \pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(f) = \pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(f)$. $\dashv$ *(Claim)*

We let $G_{\bar{A}} = \{(c, g) \in G \; : \; c \text{ is } \bar{A}\text{-principal}\}$. Then we note that $\{1_{T_{\bar{X}}}\} \times F \subseteq G$ and that $1_{T_{\bar{X}}}$ is $\bar{A}$-principal, and so $\{1_{T_{\bar{X}}}\} \times F \subseteq G_{\bar{A}}$. Therefore $\varphi_{\bar{A}}[F] \subseteq \tilde{\psi}_{\bar{A}}[G_{\bar{A}}]$. Since $\tilde{\psi}_{\bar{A}}[G_{\bar{A}}] \subseteq \tilde{\psi}_{\bar{A}}[G]$, it is enough to show that $\tilde{\psi}_{\bar{A}}[G_{\bar{A}}]$ is a Gröbner basis of $\langle \varphi_{\bar{A}}[F] \rangle$ in $T^*_{\bar{A}}[\bar{X} \setminus \bar{A}]$. Furthermore, since each $f \in \tilde{\psi}_{\bar{A}}[G_{\bar{A}}]$ is boolean closed by the previous Claim, it is enough to show that $\mathrm{SPol}(f, g) \xrightarrow{*}_{\tilde{\psi}_{\bar{A}}[G_{\bar{A}}]} 0$ for each $f, g \in \tilde{\psi}_{\bar{A}}[G_{\bar{A}}]$ by Theorem 1. So we pick two coherent pairs $(c, f), (d, g) \in G_{\bar{A}}$ arbitrarily.

If $s = \mathrm{SPol}(\pi_{\bar{A}}(c) \cdot \varphi_{\bar{A}}(g), \pi_{\bar{A}}(d) \cdot \varphi_{\bar{A}}(g)) = 0$, there is nothing to do. So we assume that $s \neq 0$ and so $\pi_{\bar{A}}(c) \cdot \pi_{\bar{A}}(d) \neq 0$. Then we have $c \cdot d \neq 0$ by Lemma 4. So, in the algorithm **GröbnerBasis**, $\mathbf{SPol}_{\bar{A}}((c, f), (d, g))$ is calculated. Say $(e, h) = \mathbf{SPol}_{\bar{A}}((c, f), (d, g)) \in T^p_{\bar{X}} \times K[\bar{X}]$. We also let $\bar{c} = \pi_{\bar{A}}(c)$ and $\bar{d} = \pi_{\bar{A}}(d)$. Then we have

$$\begin{aligned} \mathrm{SPol}(\bar{c} \cdot \varphi_{\bar{A}}(f), \bar{d} \cdot \varphi_{\bar{A}}(g)) &= \psi_{\bar{A}}(\mathbf{SPol}_{\bar{A}}((\bar{c}, f), (\bar{d}, g)) \\ &= \tilde{\psi}_{\bar{A}}(\mathbf{SPol}_{\bar{A}}((c, f), (d, g)) \\ &= \tilde{\psi}_{\bar{A}}((e, h)) \end{aligned}$$

by Lemma 4 again. So we show that $\mathbf{NormalForm}_{\bar{A}}((\pi_{\bar{A}}(e), h), \bar{G}_{\bar{A}}) = \emptyset$ where $\bar{G}_{\bar{A}} = \{(\pi_{\bar{A}}(c), g) \; : \; (c, g) \in G_{\bar{A}}\}$.

Since $\psi_{\bar{A}}[G]$ is a Gröbner basis, we know that $\mathbf{NormalForm}_{\bar{A}}((e, h), G) = \emptyset$. Though we should remind that a monomial reduction in this paper contains a split stage not only a reduction stage, we know that calculation of $\mathbf{NormalForm}_{\bar{A}}((e, h), G)$ contains at most finite many split stages because it stops. So, we first apply the all split stages contained in the calculation of $\mathbf{NormalForm}_{\bar{A}}((e, h), G)$ at the beginning, we can split e to $e_1, \ldots, e_l$ such that

1. $e_1, \ldots, e_l \in T^p_{\bar{X}}$,
2. $\mathrm{supp}(e) = \mathrm{supp}(e_1) \cup \cdots \cup \mathrm{supp}(e_l)$,
3. $\mathrm{supp}(e_i) \cap \mathrm{supp}(e_j) = \emptyset$ for $i, j = 1, \ldots, l$ with $i \neq j$, and
4. the calculation of $\mathbf{NormalForm}_{\bar{A}}((e_i, h), G) = \emptyset$ contains no split stages for each $i = 1, \ldots, l$.

Then reordering $\langle e_i : i = 1, \dots, l\rangle$, we can assume that $e_1, \dots, e_m$ are $\bar{A}$-principal and $e_{m+1}, \dots, e_l$ are not $\bar{A}$-principal.

Claim. We have the following:

1. $\mathrm{supp}(\pi_{\bar{A}}(e)) = \mathrm{supp}(\pi_{\bar{A}}(e_1)) \cup \cdots \cup \mathrm{supp}(\pi_{\bar{A}}(e_m))$, and
2. $\mathrm{supp}(\pi_{\bar{A}}(e_i)) \cap \mathrm{supp}(\pi_{\bar{A}}(e_j)) = \emptyset$ for $i, j = 1, \dots, m$ with $i \neq j$.

Proof (Claim). We let $a_1, a_2 \in T^*_{\bar{X}}$ be such that $a_1 = e_1 + \cdots + e_m$ and that $a_2 = e_{m+1} + \cdots + e_l$ in $T_{\bar{X}}$. It is easy to see that a_1 is $\bar{A}$-principal and that a_2 is not A-principal. Since $e = a_1 + a_2 \in T^p_{\bar{X}}$, we can see that $\pi_{\bar{A}}(e) = \pi_{\bar{A}}(a_1)$ by Lemma 5. So we have $\mathrm{supp}(\pi_{\bar{A}}(e)) = \mathrm{supp}(\pi_{\bar{A}}(e_1) \cup \cdots \cup \mathrm{supp}(\pi_{\bar{A}}(e_m))$.

For $1 \leq i < j \leq m$, if we had $\pi_{\bar{A}}(e_i) \cdot \pi_{\bar{A}}(e_j) \neq 0$, then we had $e_i \cdot e_j \neq 0$ by the fact e_i, e_j are $\bar{A}$-principal and Lemma 4. So we have $\mathrm{supp}(\pi_{\bar{A}}(e_i) \cap \mathrm{supp}(\pi_{\bar{A}}(e_j)) \neq 0$ for $1 \leq i < j \leq m$. $\dashv$ *(Claim)*

Thus it is enough to show that $\mathbf{NormalForm}_{\bar{A}}((\pi_{\bar{A}}(e_i), h), \bar{G}_{\bar{A}}) = \emptyset$ for each $i = 1, \dots, m$. So we fix i such that $1 \leq i \leq m$. Since we know that $\mathbf{NormalForm}_{\bar{A}}((e_i, h), G) = \emptyset$, There is a sequence of reductions

$$(e_i, h) \longrightarrow_{g_1} (e_i, h_1) \longrightarrow_{g_2} \cdots \longrightarrow_{g_r} (e_i, h_r)$$

such that $e_i \cdot \varphi_{\bar{A}}(h_r) = 0$ and that $g_1, \dots, g_r \in G$. Then we notice that $g_1, \dots, g_r$ are $\bar{A}$-principal, since no split stage is included. So we have $\pi_{\bar{A}}(g_1), \dots, \pi_{\bar{A}}(g_r) \in \bar{G}_{\bar{A}}$. Furthermore, we have $a_j, b_j \in K[\bar{A}]$ and a term β_j of $\bar{X} \setminus \bar{A}$ such that $h_{j+1} = a_j h_j + b_j \beta_j \bar{g}_{j+1}$ where $\bar{g}_{j+1} \in K[\bar{X}]$ is such that $g_{j+1} = (c_{j+1}, \bar{g}_{j+1})$ for some c_{j+1} for each $j = 1, \dots, r-1$, by the definition of **Add**. So, from this sequence, we can construct another sequence of reductions below:

$$(\pi_{\bar{A}}(e_i), h) \longrightarrow_{\pi_{\bar{A}}(g_1)} (\pi_{\bar{A}}(e_i), h_1) \longrightarrow_{\pi_{\bar{A}}(g_2)} \cdots \longrightarrow_{\pi_{\bar{A}}(g_r)} (\pi(\bar{e}_i), h_r).$$

Since we know that $e_i \cdot \varphi_{\bar{A}}(h_r) = e_i \cdot \varphi_{\bar{A}}(0)$, we have that $\pi_{\bar{A}}(e_i) \cdot \varphi_{\bar{A}}(h_r) = \pi_{\bar{A}}(e_i) \cdot \varphi_{\bar{A}}(0) = 0_{T_{\bar{A}}[\bar{X} \setminus \bar{A}]}$ by Lemma 6. Thus we have $\mathbf{NormalForm}_{\bar{A}}((\pi_{\bar{A}}(e_i), h), \bar{G}_{\bar{A}}) = \emptyset$.

So we have $\tilde{\psi}_{\bar{A}}[G]$ is a Gröbner basis of the ideal $\langle \varphi_{\bar{A}}[F]\rangle$ in $T_{\bar{A}}[\bar{X} \setminus \bar{A}]$. To see that $\tilde{\psi}_{\bar{A}}[G]$ is an ACGB for F, we refer to the Theorem 4.3 in [6]. □

Now we have the following. The proof is shown in [7].

Theorem 3. *Let F be a finite subset of $K[\bar{X}]$ and $\mathcal{A}$ be a subset of $\mathcal{P}(\bar{X})$. Let G be the finite set of coherent pairs such that $G =$* $\mathbf{GröbnerBasis}(F, \mathcal{A})$. *Let $\bar{G} = \{g \in K[\bar{X}] \ : \ \exists c \ (c, g) \in G\}$. Then $\bar{G}$ forms an $\mathcal{A}$-comprehensive Gröbner basis, i.e., $\bar{G}$ is a comprehensive Gröbner basis with respect to parameters $\bar{A}$ for any $\bar{A} \in \mathcal{A}$.*

5 Examples and Remarks

We implemented the algorithm to compute $\mathcal{A}$-comprehensive Gröbner bases in the case K is the field of the complex numbers $\mathbb{C}$ on trial. In this section, we give computational examples of our implementation.

Example 1. Fix the lex order with $X > Y$. Find a full comprehensive Gröbner basis for the finite set of polynomials $F = \{XY + 1, X^2 + 2\} \subseteq \mathbb{C}[X, Y]$.

From an input `fcgr([x*y+1,x^2+2],[[x,y],[x],[y]],2);` our program written in Risa/Asir [3] produces `[-2*y^2-1,-x+2*y,y*x+1,-x^2+4*y^2,x^2+2]`, and so a full comprehensive Gröbner basis is

$$G = \{2Y^2 + 1, X - 2Y, XY + 1, X^2 - 4Y^2, X^2 + 2\}.$$

In fact, we can check that the reduced Gröbner basis ($\emptyset$-comprehensive Gröbner basis) is $\{2Y^2+1, X-2Y\}$, $\{Y\}$-comprehensive Gröbner basis is $\{X-2Y, XY+1, X^2+2\}$, and $\{X\}$-comprehensive Gröbner basis is $\{X-2Y, XY+1, 2Y^2+1\}$. We should note that F itself is an $\{X, Y\}$-comprehensive Gröbner basis.

In this example, we remark that we should not make the base $\varphi_X[G]$ reduced in $T_{\bar{X}}[Y]$. For example, we see that $\varphi_X(2Y^2 + 1) \overset{*}{\longrightarrow}_{\varphi_X(X-2Y)} \varphi_X(X^2 + 2) \longrightarrow_{\varphi_X(X^2+2)} 0$, though $2Y^2 + 1$ is required for the base $\varphi_Y[G]$ in $T_Y[X]$. So we should take care to delete (or modify) a comprehensive pair by monomial reductions in the structure $T^{\mathfrak{p}}_{\bar{X}} \times K[\bar{X}]$, though adding a faithful comprehensive pair is safe.

Example 2. Fix the lex order with $X > Y > A > B$. Find a $\{\emptyset, \{A\}, \{B\}, \{A, B\}\}$-comprehensive Gröbner basis for the set of polynomials $F = \{(A + 1)X^2Y + B, (B^2 + 2)X^2 + AY\} \subseteq \mathbb{C}[X, Y, A, B]$.

From an input `fcgr([(a+1)*x^2*y+b,(b^2+2)*x^2+a*y],[[x,y,a,b], [x,y,a],[x,y,b],[x,y]],2);` our program outputs the following set of eight polynomials

$$\begin{gathered}\{(-a^2 - a)y^2 + b^3 + 2b, (b^2 + 2)x^2 + ay, (a + 1)yx^2 + b,\\ (b^2a - 2)yx^2 + a^2y^2 - 2b, (-b^2 - 2)x^4 + yx^2 + b, ((-b^2 - 2)a - b^2 - 2)x^4 + ba,\\ (-ba - b)yx^4 + 2x^2 + ay, ((-b^3 - 2b)a - b^3 - 2b)x^6 - 2ax^2 - a^2y\}.\end{gathered}$$

On the other hand, we can see that each $\bar{A}$-comprehensive Gröbner basis can be expressed as below where $G' = \{(-a^2 - a)y^2 + b^3 + 2b, (b^2 + 2)x^2 + ay\}$:

$\bar{A}$	$\bar{A}$-comprehensive Gröbner basis
$\emptyset$	$G' \cup \{(a + 1)yx^2 + b\}$
$\{A\}$	$G' \cup \{(a + 1)yx^2 + b, (-ba - b)yx^4 + 2x^2 + ay\}$
$\{B\}$	$G' \cup \{((-b^2 - 2)a - b^2 - 2)x^4 + ba, (-b^2 - 2)x^4 + yx^2 + b\}$
$\{A, B\}$	$G' \cup \{((-b^2 - 2)a - b^2 - 2)x^4 + ba, (a + 1)yx^2 + b\}$

Then we can see that the polynomial $((-b^3-2b)a-b^3-2b)x^6-2ax^2-a^2y$ in the output of `fcgr(...)` does not appear in the table above. So it is overplus for an $\{\emptyset,\{A\},\{B\},\{A,B\}\}$-comprehensive Gr"obner basis for F. But the present implementation does not have a sophisticated method to delete such overplus comprehensive pairs.

6 Conclusion and Remarks

In [7], we adopted the structure $T_{\bar{A}}[\bar{X}\setminus\bar{A}]$ and give an algorithm to compute $\bar{A}$-comprehensive Gröbner bases with fixing $\bar{A}\subseteq\bar{X}$ during computation. In this paper, extending the idea of it, we show an algorithm to compute $\bar{A}$-comprehensive Gröbner bases for each $\bar{A}\subseteq\bar{X}$ simultaneously. We can think this new algorithm enable to compute at most 2^n many comprehensive Gröbner bases for a given finite set of polynomials where n is the number of variables. Thus the number of polynomials which are generated as S-polynomials during the computation may be less than ones generated during the sequential computation of comprehensive Gröbner bases.

On the other hand, as we mentioned in Section 5, it is not easy to reduce the number of comprehensive pairs during computation, and so many technique to reduce amount of computation used in computation Gröbner bases cannot be applied to the algorithm in this paper. Furthermore, several methods to remove duplications of the same computation used in the algorithm of [7] is not introduced to the new algorithm, because we think that the introduction may makes the data structure expressing comprehensive pairs too complex. As a result of them, the computational speed of the present implementation is slower than sequential computation by a existing implementation in many cases. So we should try to introduce such methods to reduce computation of full comprehensive Gröbner bases.

References

1. Cox, D., Little, J. and O'Shea, D. (1996). Ideals, Varieties and Algorithms – An Introduction to Computational Algebraic Geometry and Commutative Algebra – Second Edition, Springer.
2. Montes, A. (2002). A new algorithm for discussing Gröbner basis with parameters, J. Symb. Comp. 33, 1-2, 183–208.
3. Noro, M. and Takeshima, T. (1992). Risa/Asir – A Computer Algebra System. International Symposium on Symbolic and Algebraic Computation (ISSAC 92), Proceedings, 387–396.
4. Sato, Y. and Suzuki, A. (2001). Discrete Comprehensive Gröbner Bases. International Symposium on Symbolic and Algebraic Computation (ISSAC 2001), Proceedings, 292–296.
5. Suzuki, A. and Sato, Y. (2002). An Alternative approach to Comprehensive Gröbner Bases. International Symposium on Symbolic and Algebraic Computation (ISSAC 2002), Proceedings, 255–261.

6. Suzuki, A. and Sato, Y. (2003). An Alternative approach to Comprehensive Gröbner Bases. J. Symb. Comp. 36/3-4 649–667.
7. Suzuki, A. and Sato, Y. (2004). Comprehensive Gröbner Bases via ACGB, Proceedings of the tenth international conference on Application of Computer Algebra (ACA 2004), 65–73.
8. Suzuki, A. (2005). Full Comprehensive Gröbner Bases (Extended Abstract), Proceedings of the Algorithmic Algebra and Logic 2005 (A3L 2005), 249–252.
9. Weispfenning, V. (1989). Gröbner bases in polynomial ideals over commutative regular rings, EUROCAL '87, J. H. Davenport Ed., Springer LNCS **378**, 336–347.
10. Weispfenning, V. (1992). Comprehensive Gröbner bases, J. Symb. Comp. 14/1, 1–29.
11. Weispfenning, V. (2002). Comprehensive Gröbner Bases and Regular Rings, Symposium in Honor of Bruno Buchberger's 60th Birthday (LMCS 2002) Proceedings, 256–265.

Recursive Polynomial Remainder Sequence and the Nested Subresultants

Akira Terui

Graduate School of Pure and Applied Sciences,
University of Tsukuba,
Tsukuba, 305-8571, Japan
terui@math.tsukuba.ac.jp

Abstract. We give two new expressions of subresultants, *nested subresultant* and *reduced nested subresultant*, for the recursive polynomial remainder sequence (PRS) which has been introduced by the author. The reduced nested subresultant reduces the size of the subresultant matrix drastically compared with the recursive subresultant proposed by the authors before, hence it is much more useful for investigation of the recursive PRS. Finally, we discuss usage of the reduced nested subresultant in approximate algebraic computation, which motivates the present work.

1 Introduction

The polynomial remainder sequence (PRS) is one of fundamental tools in computer algebra. Although the Euclidean algorithm (see Knuth [1] for example) for calculating PRS is simple, coefficient growth in PRS makes the Euclidean algorithm often very inefficient. To overcome this problem, the mechanism of coefficient growth has been extensively studied through the theory of subresultants; see Collins [2], Brown and Traub [3], Loos [4], etc. By the theory of subresultant, we can remove extraneous factors of the elements of PRS systematically.

In our earlier research [5], we have introduced a variation of PRS called "recursive PRS," and its subresultant called "recursive subresultant." The recursive PRS is a result of repeated calculation of PRS for the GCD and its derivative of the original PRS, until the element becomes a constant. Then, the coefficients of the elements in the recursive PRS depend on the coefficients of the initial polynomials. By the recursive subresultants, we have given an expression of the coefficients of the elements in the recursive PRS in certain determinants of coefficients of the initial polynomials. However, as the recursion depth of the recursive PRS has increased, the recursive subresultant matrices have become so large that use of them have often become impractical [6].

In this paper, we give two other expressions of subresultants for the recursive PRS, called "nested subresultant" and "reduced nested subresultant." The nested subresultant is a subresultant with expression of "nested" determinants, used to show the relationship between the recursive and the reduced nested subresultants. The reduced nested subresultant has the same form as the result of

Gaussian elimination with the Sylvester's identity on the nested subresultant, hence it reduces the size of the subresultant matrix drastically compared with the recursive subresultant. Therefore, it is much more useful than the recursive subresultant for investigation of the recursive PRS.

This paper is organized as follows. In Sect. 2, we review the concept of the recursive PRS and the recursive subresultant. In Sect. 3, we define the nested subresultant and show its equivalence to the recursive subresultant. In Sect. 4, we define the reduced nested subresultant and show that it is a reduced expression of the nested subresultant. In Sect. 5, we discuss briefly usage of the reduced nested subresultant in approximate algebraic computation.

2 Recursive PRS and Recursive Subresultants

Let R be an integral domain and K be its quotient field, and polynomials F and G be in $R[x]$. When we calculate PRS for F and G which have a nontrivial GCD, we usually stop the calculation with the GCD. However, it is sometimes useful to continue the calculation by calculating the PRS for the GCD and its derivative; this is used for square-free decompositions. We call such a PRS a "recursive PRS."

To make this paper self-contained, we briefly review the definitions and the properties of the recursive PRS and the recursive subresultant, with necessary definitions of subresultants (for detailed discussions, see Terui [5]). In this paper, we follow definitions and notations by von zur Gathen and Lücking [7].

2.1 Recursive PRS

Definition 1 (Polynomial Remainder Sequence (PRS)). *Let F and G be polynomials in $R[x]$ of degree m and n ($m > n$), respectively. A sequence $(P_1,\ldots,P_l)$ of nonzero polynomials is called a* polynomial remainder sequence (PRS) *for F and G, abbreviated to* $\mathrm{prs}(F,G)$*, if it satisfies $P_1 = F$, $P_2 = G$, $\alpha_i P_{i-2} = q_{i-1}P_{i-1} + \beta_i P_i$, for $i = 3,\ldots,l$, where $\alpha_3,\ldots,\alpha_l$, $\beta_3,\ldots,\beta_l$ are elements of R and $\deg(P_{i-1}) > \deg(P_i)$. A sequence $((\alpha_3,\beta_3),\ldots,(\alpha_l,\beta_l))$ is called a* division rule *for* $\mathrm{prs}(F,G)$*. If P_l is a constant, then the PRS is called* complete*.* □

Definition 2 (Recursive PRS). *Let F and G be the same as in Definition 1. Then, a sequence $(P_1^{(1)},\ldots,P_{l_1}^{(1)},P_1^{(2)},\ldots,P_{l_2}^{(2)},\ldots,P_1^{(t)},\ldots,P_{l_t}^{(t)})$ of nonzero polynomials is called a* recursive polynomial remainder sequence *(recursive PRS) for F and G, abbreviated to* $\mathrm{rprs}(F,G)$*, if it satisfies*

$$\begin{aligned}
&P_1^{(1)} = F,\quad P_2^{(1)} = G,\quad P_{l_1}^{(1)} = \gamma_1\cdot\gcd(P_1^{(1)},P_2^{(1)})\quad \textit{with } \gamma_1\in R,\\
&(P_1^{(1)},P_2^{(1)},\ldots,P_{l_1}^{(1)}) = \mathrm{prs}(P_1^{(1)},P_2^{(1)}),\\
&P_1^{(k)} = P_{l_{k-1}}^{(k-1)},\ P_2^{(k)} = \frac{d}{dx}P_{l_{k-1}}^{(k-1)},\ P_{l_k}^{(k)} = \gamma_k\cdot\gcd(P_1^{(k)},P_2^{(k)})\ \textit{with } \gamma_k\in R,\\
&(P_1^{(k)},P_2^{(k)},\ldots,P_{l_k}^{(k)}) = \mathrm{prs}(P_1^{(k)},P_2^{(k)}),
\end{aligned}\tag{1}$$

for $k = 2, \ldots, t$. If $\alpha_i^{(k)}$, $\beta_i^{(k)} \in R$ satisfy $\alpha_i^{(k)} P_{i-2}^{(k)} = q_{i-1}^{(k)} P_{i-1}^{(k)} + \beta_i^{(k)} P_i^{(k)}$ for $k = 1, \ldots, t$ and $i = 3, \ldots, l_k$, then a sequence $((\alpha_3^{(1)}, \beta_3^{(1)}), \ldots, (\alpha_{l_t}^{(t)}, \beta_{l_t}^{(t)}))$ is called a division rule *for* rprs(F, G). *Furthermore, if $P_{l_t}^{(t)}$ is a constant, then the recursive PRS is called complete.* □

In this paper, we use the following notations. Let $c_i^{(k)} = \mathrm{lc}(P_i^{(k)})$, $n_i^{(k)} = \deg(P_i^{(k)})$, $j_0 = m$ and $j_k = n_l^{(k)}$ for $k = 1, \ldots, t$ and $i = 1, \ldots, l_k$, and let $d_i^{(k)} = n_i^{(k)} - n_{i+1}^{(k)}$ for $k = 1, \ldots, t$ and $i = 1, \ldots, l_k - 1$.

2.2 Recursive Subresultants

We construct "recursive subresultant matrix" whose determinants represent the elements of the recursive PRS by the coefficients of the initial polynomials.

Let F and G be polynomials in $R[x]$ such that

$$F(x) = f_m x^m + \cdots + f_0 x^0, \quad G(x) = g_n x^n + \cdots + g_0 x^0, \tag{2}$$

with $m \geq n > 0$. For a square matrix M, we denote its determinant by $|M|$.

Definition 3 (Sylvester Matrix and Subresultant Matrix). *Let F and G be as in* (2). *The* Sylvester matrix *of F and G, denoted by $N(F, G)$ in (3), is an $(m+n) \times (m+n)$ matrix constructed from the coefficients of F and G. For $j < n$, the j-th* subresultant matrix *of F and G, denoted by $N^{(j)}(F, G)$ in (3), is an $(m+n-j) \times (m+n-2j)$ sub-matrix of $N(F, G)$ obtained by taking the left $n - j$ columns of coefficients of F and the left $m - j$ columns of coefficients of G.*

$$N(F,G) = \underbrace{\begin{pmatrix} f_m & & & g_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ f_0 & & f_m & g_0 & & g_n \\ & \ddots & \vdots & & \ddots & \vdots \\ & & f_0 & & & g_0 \end{pmatrix}}_{n \qquad\qquad m}, \quad N^{(j)}(F,G) = \underbrace{\begin{pmatrix} f_m & & & g_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ f_0 & & f_m & g_0 & & g_n \\ & \ddots & \vdots & & \ddots & \vdots \\ & & f_0 & & & g_0 \end{pmatrix}}_{n-j \qquad\qquad m-j}. \tag{3}$$

Definition 4 (Recursive Subresultant Matrix). *Let F and G be defined as in* (2)*, and let $(P_1^{(1)}, \ldots, P_{l_1}^{(1)}, \ldots, P_1^{(t)}, \ldots, P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 2. Then, for each tuple of numbers (k, j) with $k = 1, \ldots, t$ and $j = j_{k-1} - 2, \ldots, 0$, define matrix $\bar{N}^{(k,j)} = \bar{N}^{(k,j)}(F, G)$ recursively as follows.*

1. *For $k = 1$, let $\bar{N}^{(1,j)}(F, G) = N^{(j)}(F, G)$.*
2. *For $k > 1$, let $\bar{N}^{(k,j)}(F, G)$ consist of the upper block and the lower block, defined as follows:*
   (a) The upper block is partitioned into $(j_{k-1} - j_k - 1) \times (j_{k-1} - j_k - 1)$ blocks with diagonal blocks filled with $\bar{N}_U^{(k-1, j_{k-1})}$, where $\bar{N}_U^{(k-1, j_{k-1})}$ is a sub-matrix of $\bar{N}^{(k-1, j_{k-1})}(F, G)$ obtained by deleting the bottom $j_{k-1} + 1$ rows.

(b) Let $\bar{N}_L^{(k-1,j_{k-1})}$ be a sub-matrix of $\bar{N}^{(k-1,j_{k-1})}$ obtained by taking the bottom $j_{k-1}+1$ rows, and let $\bar{N}_L^{\prime(k-1,j_{k-1})}$ be a sub-matrix of $\bar{N}_L^{(k-1,j_{k-1})}$ by multiplying the $(j_{k-1}+1-\tau)$-th rows by τ for $\tau = j_{k-1},\dots,1$, then by deleting the bottom row. Then, the lower block consists of $j_{k-1}-j-1$ blocks of $\bar{N}_L^{(k-1,j_{k-1})}$ such that the leftmost block is placed at the top row of the container block and the right-side block is placed down by 1 row from the left-side block, then followed by $j_{k-1}-j$ blocks of $\bar{N}_L^{\prime(k-1,j_{k-1})}$ placed by the same manner as $\bar{N}_L^{(k-1,j_{k-1})}$.

Readers can find the structures of $\bar{N}^{(k,j)}(F,G)$ in the figures in Terui [5]. Then, $\bar{N}^{(k,j)}(F,G)$ is called the (k,j)-th recursive subresultant matrix *of F and G.* □

Proposition 1. *For $k = 1,\dots,t$ and $j < j_{k-1}-1$, the numbers of rows and columns of $\bar{N}^{(k,j)}(F,G)$, the (k,j)-th recursive subresultant matrix of F and G are $(m+n-2j_1)\left\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\right\}(2j_{k-1}-2j-1)+j$, $(m+n-2j_1)\left\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\right\}(2j_{k-1}-2j-1)$, respectively.* □

Definition 5 (Recursive Subresultant). *Let F and G be defined as in* (2)*, and let $(P_1^{(1)},\dots,P_{l_1}^{(1)},\dots,P_1^{(t)},\dots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 2. For $j = j_{k-1}-2,\dots,0$ and $\tau = j,\dots,0$, let $\bar{N}_\tau^{(k,j)} = M_\tau^{(k,j)}(F,G)$ be a sub-matrix of the (k,j)-th recursive subresultant matrix $\bar{N}^{(k,j)}(F,G)$ obtained by taking the top $(m+n-2j_1)\times\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\}(2j_{k-1}-2j-1)-1$ rows and the $((m+n-2j_1)\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\}(2j_{k-1}-2j-1)+j-\tau)$-th row (note that $\bar{N}_\tau^{(k,j)}$ is a square matrix). Then, the polynomial $\bar{\mathrm{S}}_{k,j}(F,G) = |\bar{N}_j^{(k,j)}|x^j+\cdots+|\bar{N}_0^{(k,j)}|x^0$ is called the (k,j)-th* recursive subresultant *of F and G.* □

3 Nested Subresultants

Although the recursive subresultant can represent the coefficients of the elements in recursive PRS, the size of the recursive subresultant matrix becomes larger rapidly as the recursion depth of the recursive PRS becomes deeper, hence making use of the recursive subresultant matrix become more inefficient.

To overcome this problem, we introduce other representations for the subresultant which is equivalent to the recursive subresultant up to a constant, and more efficient to calculate. The nested subresultant matrix is a subresultant matrix whose elements are again determinants of certain subresultant matrices (or even the nested subresultant matrices), and the nested subresultant is a subresultant whose coefficients are determinants of the nested subresultant matrices.

In this paper, the nested subresultant is mainly used to show the relationship between the recursive subresultant and the reduced nested subresultant, which is defined in the next section.

Definition 6 (Nested Subresultant Matrix). *Let F and G be defined as in (2), and let $(P_1^{(1)},\dots,P_{l_1}^{(1)},\dots,P_1^{(t)},\dots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 2. Then, for each tuple of numbers (k,j) with $k=1,\dots,t$ and $j=j_{k-1}-2,\dots,0$, define matrix $\tilde{N}^{(k,j)}(F,G)$ recursively as follows.*

1. *For $k=1$, let $\tilde{N}^{(1,j)}(F,G)=N^{(j)}(F,G)$.*
2. *For $k>1$ and $\tau=0,\dots,j_{k-1}$, let $\tilde{N}_\tau^{(k-1,j_{k-1})}$ be a sub-matrix of $\tilde{N}^{(k-1,j_{k-1})}$ by taking the top $(n_1^{(k-1)}+n_2^{(k-1)}-2j_{k-1}-1)$ rows and the $(n_1^{(k-1)}+n_2^{(k-1)}-j_{k-1}-\tau)$-th row (note that $\tilde{N}_\tau^{(k-1,j_{k-1})}$ is a square matrix). Now, let*
$$\tilde{N}^{(k,j)}(F,G)=N^{(j)}\left(\tilde{\mathrm{S}}_{k-1,j_{k-1}}(F,G),\frac{d}{dx}\tilde{\mathrm{S}}_{k-1,j_{k-1}}(F,G)\right), \qquad (4)$$
*where $\tilde{\mathrm{S}}_{k-1,j_{k-1}}(F,G)$ is defined by Definition 7. Then, $\tilde{N}^{(k,j)}(F,G)$ is called the (k,j)-*th nested subresultant matrix *of F and G.* □

Definition 7 (Nested Subresultant). *Let F and G be defined as in (2), and let $(P_1^{(1)},\dots,P_{l_1}^{(1)},\dots,P_1^{(t)},\dots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 2. For $j=j_{k-1}-2,\dots,0$ and $\tau=j,\dots,0$, let $\tilde{N}_\tau^{(k,j)}=\tilde{N}_\tau^{(k,j)}(F,G)$ be a sub-matrix of the (k,j)-th nested recursive subresultant matrix $\tilde{N}^{(k,j)}(F,G)$ obtained by taking the top $n_1^{(k)}+n_2^{(k)}-2j-1$ rows and the $(n_1^{(k)}+n_2^{(k)}-j-\tau)$-th row (note that $\tilde{N}_\tau^{(k,j)}$ is a square matrix). Then, the polynomial $\tilde{\mathrm{S}}_{k,j}(F,G)=|\tilde{N}_j^{(k,j)}|x^j+\cdots+|\tilde{N}_0^{(k,j)}|x^0$ is called the (k,j)-*th nested subresultant *of F and G.* □

We show that the nested subresultant is equal to the recursive subresultant up to a sign.

Theorem 1. *Let F and G be defined as in (2), and let $(P_1^{(1)},\dots,P_{l_1}^{(1)},\dots,P_1^{(t)},\dots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 2. For $k=2,\dots,t$ and $j=j_{k-1}-2,\dots,0$, define $u_{k,j}$, $b_{k,j}$, $r_{k,j}$ and R_k as follows: let $u_{k,j}=(m+n-2j_1)\left\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\right\}(2j_{k-1}-2j-1)$ with $u_k=u_{k,j_k}$ and $u_1=m+n-2j_1$, $b_{k,j}=2j_{k-1}-2j-1$ with $b_k=b_{k,j_k}$ and $b_1=1$, $r_{k,j}=(-1)^{(u_{k-1}-1)(1+2+\cdots+(b_{k,j}-1))}$ with $r_k=r_{k,j_k}$ and $r_{1,j}=1$ for $j<n$, and $R_k=(R_{k-1})^{b_k}r_k$ with $R_0=R_1=1$. Then, we have*
$$\tilde{\mathrm{S}}_{k,j}(F,G)=(R_{k-1})^{b_{k,j}}\,r_{k,j}\,\mathrm{S}_{k,j}(F,G). \qquad (5)$$

To prove Theorem 1, we prove the following lemma.

Lemma 1. *For $k=1,\dots,t$, $j=j_{k-1}-2,\dots,0$ and $\tau=j,\dots,0$, we have*
$$|\tilde{N}_\tau^{(k,j)}(F,G)|=(R_{k-1})^{b_{k,j}}\,r_{k,j}\,|\bar{N}_\tau^{(k,j)}(F,G)|. \qquad (6)$$

Proof. By induction on k. For $k=1$, it is obvious by the definitions of the recursive and the nested subresultants. Assume that the lemma is valid for $1,\dots,k-1$.

Then, for $\tau = j_{k-1}, \dots, 0$, we have $|\tilde{N}_\tau^{(k-1,j_{k-1})}| = R_{k-1}|\bar{N}_\tau^{(k-1,j_{k-1})}|$. For an element in recursive PRS $P_i^{(k)}(x)$, expressed as $P_i^{(k)}(x) = a_{i,n_i^{(k)}}^{(k)} x^{n_i^{(k)}} + \cdots + a_{i,0}^{(k)} x^0$, denote the coefficient vector for $P_i^{(k)}(x)$ by $\boldsymbol{p}_i^{(k)} = {}^t(a_{i,n_i^{(k)}}^{(k)}, \dots, a_{i,0}^{(k)})$. Then, there exist certain eliminations and exchanges on columns which transform $\tilde{N}^{(k-1,j_{k-1})}$ to $\tilde{M}^{(k-1,j_{k-1})} = \left(\begin{array}{c|c} \tilde{W}_{k-1} & O \\ \hline * & \boldsymbol{p}_{l_{k-1}}^{(k-1)} \end{array}\right) = \left(\begin{array}{c|c} \tilde{W}_{k-1} & O \\ \hline * & \boldsymbol{p}_1^{(k)} \end{array}\right)$, such that, for $\tau = j, \dots, 0$, we have $\tilde{M}_\tau^{(k-1,j_{k-1})} = \left(\begin{array}{c|c} \tilde{W}_{k-1} & O \\ \hline * & a_{l_{k-1},\tau}^{(k-1)} \end{array}\right) = \left(\begin{array}{c|c} \tilde{W}_{k-1} & O \\ \hline * & a_{1,\tau}^{(k)} \end{array}\right)$ with $|\tilde{M}_\tau^{(k-1,j_{k-1})}| = |\tilde{W}_{k-1}| a_{1,\tau}^{(k)}$, where $\tilde{M}_\tau^{(k-1,j_{k-1})}$ is a sub-matrix of $\tilde{M}^{(k-1,j_{k-1})}$ by taking the top $n_1^{(k-1)} + n_2^{(k-1)} - 2j_{k-1} - 1$ rows and the $(n_1^{(k-1)} + n_2^{(k-1)} - j_{k-1} - \tau)$-th row (note that the matrix $\tilde{W}_{k-1}$ is a square matrix of order $n_1^{(k-1)} + n_2^{(k-1)} - 2j_{k-1} - 1$). By the definition of $\tilde{N}^{(k,j)}(F,G)$, we have $\tilde{N}^{(k,j)}(F,G) = |\tilde{W}_{k-1}| N^{(j)}(P_1^{(k)}, P_2^{(k)})$, hence we have

$$|\tilde{N}_\tau^{(k,j)}(F,G)| = |\tilde{W}_{k-1}|^{n_1^{(k)} + n_2^{(k)} - 2j} |N_\tau^{(j)}(P_1^{(k)}, P_2^{(k)})|. \tag{7}$$

On the other hand, there exist similar transformation which transforms $\bar{N}^{(k-1,j_{k-1})}$ and $\left(\begin{array}{c} \bar{N}_U^{(k-1,j_{k-1})} \\ \hline \bar{N}_L'^{(k-1,j_{k-1})} \end{array}\right)$ into $\bar{M}^{(k-1,j_{k-1})} = \left(\begin{array}{c|c} \bar{W}_{k-1} & O \\ \hline * & \boldsymbol{p}_{l_{k-1}}^{(k-1)} \end{array}\right) = \left(\begin{array}{c|c} \bar{W}_{k-1} & O \\ \hline * & \boldsymbol{p}_1^{(k)} \end{array}\right)$ and $\bar{M}'^{(k-1,j_{k-1})} = \left(\begin{array}{c|c} \bar{W}_{k-1} & O \\ \hline * & \boldsymbol{p}_2^{(k)} \end{array}\right)$, respectively, with $|\bar{W}_{k-1}| = |\tilde{W}_{k-1}|$ by assumption. Therefore, by exchanges of columns after the above transformations on each column blocks (see Terui [5] for detail), we have

$$\begin{aligned} (R_{k-1})^{b_{k,j}}\, r_{k,j}\, |\bar{N}_\tau^{(k,j)}(F,G)| &= |\bar{W}_{k-1}|^{n_1^{(k)} + n_2^{(k)} - 2j} |N_\tau^{(j)}(P_1^{(k)}, P_2^{(k)})| \\ &= |\tilde{W}_{k-1}|^{n_1^{(k)} + n_2^{(k)} - 2j} |N_\tau^{(j)}(P_1^{(k)}, P_2^{(k)})| \\ &= |\tilde{N}_\tau^{(k,j)}(F,G)|, \end{aligned} \tag{8}$$

which proves the lemma. □

4 Reduced Nested Subresultants

The nested subresultant matrix has "nested" representation of subresultant matrices, which makes practical use difficult. However, in some cases, by Gaussian elimination of the matrix with the Sylvester's identity after some precomputations, we can reduce the representation of the nested subresultant matrix to "flat" representation, or a representation without nested determinants; this is the reduced nested subresultant (matrix). As we will see, the size of the reduced nested subresultant matrix becomes much smaller than that of the recursive subresultant matrix.

First, we show the Sylvester's identity (see also Bariess [8]), then explain the idea of reduction of the nested subresultant matrix with the Sylvester's identity by an example.

Lemma 2 (The Sylvester's Identity). *Let $A = (a_{ij})$ be $n\times n$ matrix, and, for $k = 1,\dots,n-1$, $i = k+1,\dots,n$ and $j = k+1,\dots,n$, let* $a_{i,j}^{(k)} = \begin{vmatrix} a_{11} & \cdots & a_{1k} & a_{1j} \\ \vdots & & \vdots & \vdots \\ a_{k1} & \cdots & a_{kk} & a_{kj} \\ a_{i1} & \cdots & a_{ik} & a_{ij} \end{vmatrix}$.

Then, we have $|A| \left(a_{kk}^{(k-1)}\right)^{n-k-1} = \begin{vmatrix} a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ \vdots & & \vdots \\ a_{n,k+1}^{(k)} & \cdots & a_{n,n}^{(k)} \end{vmatrix}$. □

Example 1. Let $F(x)$ and $G(x)$ be defined as

$$F(x) = a_6x^6 + a_5x^5 + \cdots + a_0, \quad G(x) = b_5x^5 + b_4x^4 + \cdots + b_0, \tag{9}$$

with $a_6 \neq 0$ and $b_5 \neq 0$. We assume that vectors of coefficients (a_6, a_5) and (b_5, b_4) are linearly independent as vectors over K, and that $\mathrm{prs}(F,G) = (P_1^{(1)} = F,\ P_2^{(1)} = G,\ P_3^{(1)} = \gcd(F,G))$ with $\deg(P_3^{(1)}) = 4$. Consider the $(2,2)$-th nested subresultant; its matrix is defined as

$$\tilde{N}^{(2,2)} = \begin{pmatrix} A_4 & 4A_4 & \\ A_3 & 3A_3 & 4A_4 \\ A_2 & 2A_2 & 3A_3 \\ A_1 & A_1 & 2A_2 \\ A_0 & & A_1 \end{pmatrix}, \quad A_j = \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ a_j & b_{j-1} & b_j \end{vmatrix}, \tag{10}$$

for $j \le 4$ with $b_j = 0$ for $j < 0$. Now, let us calculate the leading coefficient of $\tilde{\mathrm{S}}_{2,2}(F,G)$ as

$$|\tilde{N}_2^{(2,2)}| = \begin{vmatrix} A_4 & 4A_4 & \\ A_3 & 3A_3 & 4A_4 \\ A_2 & 2A_2 & 3A_3 \end{vmatrix} = \begin{vmatrix} \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ a_4 & b_3 & b_4 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 4a_4 & 4b_3 & 4b_4 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 0a_4 & 0b_3 & 0b_4 \end{vmatrix} \\ \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ a_3 & b_2 & b_3 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 3a_3 & 3b_2 & 3b_3 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 4a_4 & 4b_3 & 4b_4 \end{vmatrix} \\ \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ a_2 & b_1 & b_2 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 2a_2 & 2b_1 & 2b_2 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 3a_3 & 3b_2 & 3b_3 \end{vmatrix} \end{vmatrix} \tag{11}$$

$$= |H| = \left|(H_{p,q})\right|.$$

To apply the Sylvester's identity on H, we make the $(3,1)$ and the $(3,2)$ elements in $H_{p,2}$ and $H_{p,3}$ $(p = 1,2,3)$ equal to those elements in $H_{p,1}$, respectively, by adding the first and the second rows, multiplied by certain numbers, to the third row. For example, in $H_{1,2}$, calculate x_{12} and y_{12} by solving a system of linear equations

$$\begin{cases} a_6x_{12} + a_5y_{12} = -4a_4 + a_4 = -3a_4 \\ b_5x_{12} + b_4y_{12} = -4b_3 + b_3 = -3b_3 \end{cases}, \tag{12}$$

(Note that (12) has a solution in K by assumption), then add the first row multiplied by x_{12} and the second row multiplied by y_{12}, respectively, to the third row. Then, we have $H_{1,2} = \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ a_4 & b_3 & h_{12} \end{vmatrix}$ with $h_{12} = 4b_4 + y_{12}b_5$. Doing similar calculations for the other $H_{p,q}$, we calculate $h_{p,q}$ for $H_{p,q}$ similarly as in the above. Finally, by the Sylvester's identity, we have

$$|\tilde{N}_2^{(2,2)}| = \begin{vmatrix} a_6 & b_5 \\ a_5 & b_4 \end{vmatrix}^2 \begin{vmatrix} a_6 & b_5 & & & \\ a_5 & b_4 & b_5 & b_5 & b_5 \\ a_4 & b_3 & b_4 & h_{12} & h_{13} \\ a_3 & b_2 & b_3 & h_{22} & h_{23} \\ a_2 & b_1 & b_2 & h_{32} & h_{33} \end{vmatrix} = \begin{vmatrix} a_6 & b_5 \\ a_5 & b_4 \end{vmatrix}^2 |\hat{N}_2^{(2,2)}|, \tag{13}$$

note that we have derived $\hat{N}_2^{(2,2)}$ as a reduced form of $\tilde{N}_2^{(2,2)}$. □

Definition 8 (Reduced Nested Subresultant Matrix). *Let F and G be defined as in (2), and let $(P_1^{(1)}, \ldots, P_{l_1}^{(1)}, \ldots, P_1^{(t)}, \ldots, P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 2. Then, for each tuple of numbers (k, j) with $k = 1, \ldots, t$ and $j = j_{k-1} - 2, \ldots, 0$, define matrix $\hat{N}^{(k,j)}(F, G)$ recursively as follows.*

1. *For $k = 1$, let $\hat{N}^{(1,j)}(F, G) = N^{(j)}(F, G)$.*
2. *For $k > 1$, let $\hat{N}_U^{(k-1,j_{k-1})}(F, G)$ be a sub-matrix of $\hat{N}^{(k-1,j_{k-1})}(F, G)$ by deleting the bottom $j_{k-1} + 1$ rows, and $\hat{N}_L^{(k-1,j_{k-1})}(F, G)$ be a sub-matrix of $\hat{N}^{(k-1,j_{k-1})}(F, G)$ by taking the bottom $j_{k-1} + 1$ rows, respectively. For $\tau = j_{k-1}, \ldots, 0$ let $\hat{N}_\tau^{(k-1,j_{k-1})}(F, G)$ be a sub-matrix of $\hat{N}^{(k-1,j_{k-1})}(F, G)$ by putting $\hat{N}_U^{(k-1,j_{k-1})}(F, G)$ on the top and the $(j_{k-1} - \tau + 1)$-th row of $\hat{N}_L^{(k-1,j_{k-1})}(F, G)$ in the bottom row. Let $\hat{A}_\tau^{(k-1)} = |\hat{N}_\tau^{(k-1,j_{k-1})}|$ and construct a matrix H as*

$$H = (H_{p,q}) = N^{(j)} \left(\hat{A}^{(k-1)}(x), \frac{d}{dx} \hat{A}^{(k-1)}(x) \right), \tag{14}$$

where $\hat{A}^{(k-1)}(x) = \hat{A}_{j_{k-1}}^{(k-1)} x^{j_{k-1}} + \cdots + \hat{A}_0^{(k-1)} x^0$. Since $\hat{N}_\tau^{(k-1,j_{k-1})}$ consists of $\hat{N}_U^{(k-1,j_{k-1})}$ and a row vector in the bottom, we express $\hat{N}_U^{(k-1,j_{k-1})} = \left(U^{(k)} | \boldsymbol{v}^{(k)}\right)$, where $U^{(k)}$ is a square matrix and $\boldsymbol{v}^{(k)}$ is a column vector, and the row vector by $\left(\boldsymbol{b}_{p,q}^{(k)} \;\middle|\; g_{p,q}^{(k)}\right)$, where $\boldsymbol{b}_{p,q}^{(k)}$ is a row vector and $g_{p,q}^{(k)}$ is a number, respectively, such that

$$H_{p,q} = \left| \begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}_{p,q}^{(k)} & g_{p,q}^{(k)} \end{array} \right|, \tag{15}$$

with $\boldsymbol{b}_{p,q}^{(k)} = \mathbf{0}$ and $g_{p,q}^{(k)} = 0$ for $H_{p,q} = 0$. Furthermore, we assume that $U^{(k)}$ is not singular. Then, for $p = 1, \ldots, n_1^{(k)} + n_2^{(k)} - j$ and $q = 2, \ldots, n_1^{(k)} + n_2^{(k)} - j$, calculate a row vector $\boldsymbol{x}_{p,q}^{(k)}$ as a solution of the equation $\boldsymbol{x}_{p,q}^{(k)} U^{(k)} = \boldsymbol{b}_{p,1}^{(k)}$, and define $h_{p,q}^{(k)}$ as $h_{p,q}^{(k)} = \boldsymbol{x}_{p,q}^{(k)} \boldsymbol{v}^{(k)}$. Finally, define $\hat{N}^{(k,j)}(F, G)$ as

$$\hat{N}^{(k,j)}(F,G) = \begin{pmatrix} U^{(k)} & \boldsymbol{v}^{(k)} & \boldsymbol{v}^{(k)} & \cdots & \boldsymbol{v}^{(k)} \\ \boldsymbol{b}_{1,1}^{(k)} & g_{1,1}^{(k)} & h_{1,2}^{(k)} & \cdots & h_{1,J_{k,j}}^{(k)} \\ \boldsymbol{b}_{2,1}^{(k)} & g_{2,1}^{(k)} & h_{2,2}^{(k)} & \cdots & h_{2,J_{k,j}}^{(k)} \\ \vdots & \vdots & \vdots & & \vdots \\ \boldsymbol{b}_{I_{k,j},1}^{(k)} & g_{I_{k,j},1}^{(k)} & h_{I_{k,j},2}^{(k)} & \cdots & h_{I_{k,j},J_{k,j}}^{(k)} \end{pmatrix}, \tag{16}$$

$$\begin{aligned} I_{k,j} &= n_1^{(k)} + n_2^{(k)} - j = (2j_{k-1} - 2j - 1) + j, \\ J_{k,j} &= n_1^{(k)} + n_2^{(k)} - 2j = 2j_{k-1} - 2j - 1. \end{aligned} \tag{17}$$

Then, $\hat{N}^{(k,j)}(F,G)$ is called the (k,j)-th reduced nested subresultant matrix of F and G. □

Proposition 2. *For $k = 1,\ldots,t$ and $j < j_{k-1} - 1$, the numbers of rows and columns of the (k,j)-th reduced nested subresultant matrix $\hat{N}^{(k,j)}(F,G)$ are $(m+n-2(k-1)-2j)+j$ and $(m+n-2(k-1)-2j)$, respectively.*

Proof. By induction on k. It is obvious for $k = 1$. Assume that the proposition is valid for $1,\ldots,k-1$. Then, the numbers of rows and columns of matrix $U^{(k)}$ in (16) are equal to $(m+n-2\{(k-1)-1\}-2j_{k-1})-1$, respectively. Therefore, by (16) and (17), we prove the proposition for k. □

Note that, as Proposition 2 shows, the size of the reduced nested subresultant matrix, which is at most the sum of the degree of the initial polynomials, is much smaller than that of the recursive subresultant matrix (see Proposition 1).

Definition 9 (Reduced Nested Subresultant). *Let F and G be defined as in (2), and let $(P_1^{(1)},\ldots,P_{l_1}^{(1)},\ldots,P_1^{(t)},\ldots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 2. For $j = j_{k-1} - 2,\ldots,0$ and $\tau = j,\ldots,0$, let $\hat{N}_\tau^{(k,j)} = \hat{N}_\tau^{(k,j)}(F,G)$ be a sub-matrix of the (k,j)-th reduced nested subresultant matrix $\hat{N}^{(k,j)}(F,G)$ obtained by the top $m+n-2(k-1)-2j-1$ rows and the $(m+n-2(k-1)-j-\tau)$-th row (note that $\hat{N}_\tau^{(k,j)}(F,G)$ is a square matrix). Then, the polynomial $\hat{\mathrm{S}}_{k,j}(F,G) = |\hat{N}_j^{(k,j)}(F,G)|x^j + \cdots + |\hat{N}_0^{(k,j)}(F,G)|x^0$ is called the (k,j)-th reduced nested subresultant of F and G.* □

Now, we derive the relationship between the nested and the reduced nested subresultants.

Theorem 2. *Let F and G be defined as in (2), and let $(P_1^{(1)},\ldots,P_{l_1}^{(1)},\ldots,P_1^{(t)},\ldots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 2. For $k = 2,\ldots,t$, $j = j_{k-1} - 2,\ldots,0$ with $J_{k,j}$ as in (16), define $\hat{B}_{k,j}$ and $\hat{R}_k$ as $\hat{B}_{k,j} = |U^{(k)}|^{J_{k,j}-1}$ with $\hat{B}_k = \hat{B}_{k,j_k}$ and $\hat{B}_1 = \hat{B}_2 = 1$, and $\hat{R}_k = (\hat{R}_{k-1}\cdot\hat{B}_{k-1})^{J_{k,j_k}}$ with $\hat{R}_1 = \hat{R}_2 = 1$, respectively. Then, we have*

$$\tilde{\mathrm{S}}_{k,j}(F,G) = (\hat{R}_{k-1}\cdot\hat{B}_{k-1})^{J_{k,j}}\hat{B}_{k,j}\cdot\hat{\mathrm{S}}_{k,j}(F,G). \tag{18}$$

To prove Theorem 2, we prove the following lemma.

Lemma 3. *For $k = 1, \dots, t$, $j = j_{k-1} - 2, \dots, 0$ and $\tau = j, \dots, 0$, we have*

$$|\tilde{N}_\tau^{(k,j)}(F,G)| = (\hat{R}_{k-1} \cdot \hat{B}_{k-1})^{J_{k,j}} \hat{B}_{k,j} |\hat{N}_\tau^{(k,j)}(F,G)|. \tag{19}$$

Proof. By induction on k. For $k = 1$, it is obvious from the definitions of the nested and the reduced nested subresultants. Assume that the lemma is valid for $1, \dots, k-1$. Then, for $\tau = j_{k-1}, \dots, 0$, we have

$$\begin{aligned} |\tilde{N}_\tau^{(k-1,j_{k-1})}(F,G)| =& (\hat{R}_{k-2} \cdot \hat{B}_{k-2})^{J_{k-1,j_{k-1}}} \hat{B}_{k-1,j_{k-1}} |\hat{N}_\tau^{(k-1,j_{k-1})}(F,G)| \\ =& (\hat{R}_{k-1} \cdot \hat{B}_{k-1}) |\hat{N}_\tau^{(k-1,j_{k-1})}(F,G)|. \end{aligned} \tag{20}$$

Let $\tilde{A}_\tau^{(k-1)} = |\tilde{N}_\tau^{(k-1,j_{k-1})}|$ and $\hat{A}_\tau^{(k-1)} = |\hat{N}_\tau^{(k-1,j_{k-1})}|$. Then, by the definition of the (k,j)-th nested subresultant, we have

$$|\tilde{N}_\tau^{(k,j)}| = \left|\begin{array}{cccccc} \tilde{A}_{j_{k-1}}^{(k-1)} & & & j_{k-1}\tilde{A}_{j_{k-1}}^{(k-1)} & & \\ \vdots & \ddots & & \vdots & \ddots & \\ \vdots & & \tilde{A}_{j_{k-1}}^{(k-1)} & \vdots & & j_{k-1}\tilde{A}_{j_{k-1}}^{(k-1)} \\ \vdots & & \vdots & \vdots & & \vdots \\ \tilde{A}_{2j-j_{k-1}+3}^{(k-1)} & \cdots & \tilde{A}_{j+1}^{(k-1)} & (2j - j_{k-1} + 3)\tilde{A}_{2j-j_{k-1}+3}^{(k-1)} & \cdots & (j+2)\tilde{A}_{j+2}^{(k-1)} \\ \hline \tilde{A}_{j-j_{k-1}+\tau+2}^{(k-1)} & \cdots & \tilde{A}_\tau^{(k-1)} & (j - j_{k-1} + \tau + 2)\tilde{A}_{j-j_{k-1}+\tau+2}^{(k-1)} & \cdots & (\tau+1)\tilde{A}_{\tau+1}^{(k-1)} \end{array}\right| \tag{21}$$

$$= (\hat{R}_{k-1} \cdot \hat{B}_{k-1})^{J_{k,j}} |H'|, \tag{22}$$

where $\tilde{A}_l^{(k-1)} = 0$ for $l < 0$ and $H' = \left(H'_{p,q}\right)$ is defined as (21) with $\tilde{A}_l^{(k-1)}$ replaced by $\hat{A}_l^{(k-1)}$ (note that $\tilde{N}_\tau^{(k,j)}$ and H' are square matrices of order $J_{k,j}$). Then, by Definition 8, we can express $H'_{p,q}$ as $H'_{p,q} = \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{p,q} & g'^{(k)}_{p,q} \end{array}\right|$ with $\boldsymbol{b}'^{(k)}_{p,q} = \boldsymbol{0}$ and $g'^{(k)}_{p,q} = 0$ for $H'_{p,q} = 0$. Note that, for $q = 1, \dots, J_{k,j}$, we have $\boldsymbol{b}'^{(k)}_{p,q} = \boldsymbol{b}^{(k)}_{p,q}$ and $g'^{(k)}_{p,q} = g^{(k)}_{p,q}$ for $p = 1, \dots, J_{k,j} - 1$, and $\boldsymbol{b}'^{(k)}_{J_{k,j},q} = \boldsymbol{b}^{(k)}_{I_{k,j}-\tau,q}$ and $g'^{(k)}_{J_{k,j},q} = g^{(k)}_{I_{k,j}-\tau,q}$, where $\boldsymbol{b}^{(k)}_{p,q}$ and $g^{(k)}_{p,q}$ are defined in (15), respectively. Furthermore, by the definition of $h^{(k)}_{p,q}$ in Definition 8, we have

$$|H'| = \left|\begin{array}{cccc} \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{1,1} & g^{(k)}_{1,1} \end{array}\right| & \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{1,1} & h^{(k)}_{1,2} \end{array}\right| & \cdots & \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{1,1} & h^{(k)}_{1,J_{k,j}} \end{array}\right| \\ \vdots & \vdots & & \vdots \\ \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{J_{k,j}-1,1} & g^{(k)}_{J_{k,j}-1,1} \end{array}\right| & \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{J_{k,j}-1,1} & h^{(k)}_{J_{k,j}-1,2} \end{array}\right| & \cdots & \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{J_{k,j}-1,1} & h^{(k)}_{J_{k,j}-1,J_{k,j}} \end{array}\right| \\ \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{I_{k,j}-\tau,1} & g^{(k)}_{I_{k,j}-\tau,1} \end{array}\right| & \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{I_{k,j}-\tau,1} & h^{(k)}_{I_{k,j}-\tau,2} \end{array}\right| & \cdots & \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \boldsymbol{b}'^{(k)}_{I_{k,j}-\tau,1} & h^{(k)}_{I_{k,j}-\tau,J_{k,j}} \end{array}\right| \end{array}\right|. \tag{23}$$

By Lemma 2, we have

$$|H'| = |U^{(k)}|^{J_{k,j}-1} |\hat{N}_\tau^{(k,j)}(F,G)| = \hat{B}_{k,j} |\hat{N}_\tau^{(k,j)}(F,G)|, \tag{24}$$

hence, by putting (24) into (22), we prove the lemma. □

Remark 1. We can estimate arithmetic computing time for the (k, j)-th reduced nested resultant matrix $\hat{N}^{(k,j)}$ in (16), as follows. The computing time for the elements $h_{p,q}$ is dominated by the time for Gaussian elimination of $U^{(k)}$. Since the order of $U^{(k)}$ is equal to $m + n - 2(k-2) - 2j_{k-1}$ (see Proposition 2), it is bounded by $O((m + n - 2(k-2) - 2j_{k-1})^3)$, or $O((m + n)^3)$ (see Golub and van Loan [9] for example). We can calculate $\hat{N}^{(k,j)}(F, G)$ for $j < j_{k-1} - 2$ by $\hat{N}^{(k,0)}(F, G)$, hence the total computing time for $\hat{N}^{(k,j)}$ for the entire recursive PRS $(k = 1, \ldots, t)$ is bounded by $O(t(m+n)^3)$ (see also for the conclusion). □

5 Conclusion and Motivation

In this paper, we have given two new expressions of subresultants for the recursive PRS, the nested subresultant and the reduced nested subresultant. We have shown that the reduced nested subresultant matrix reduces the size of the matrix drastically to at most the sum of the degree of the initial polynomials compared with the recursive subresultant matrix. We have also shown that we can calculate the reduced nested subresultant matrix by solving certain systems of linear equations of order at most the sum of the degree of the initial polynomials.

A main limitation of the reduced nested subresultant in this paper is that we cannot calculate its matrix in the case the matrix $U^{(k)}$ in (15) is singular. We need to develop a method to calculate the reduced nested subresultant matrix in the case such that $U^{(k)}$ is singular in general.

From a point of view of computational complexity, the algorithm for the reduced nested subresultant matrix has a cubic complexity bound in terms of the degree of the input polynomials (see Remark 1). However, subresultant algorithms which have a quadratic complexity bound in terms of the degree of the input polynomials have been proposed ([10], [11]); the algorithms exploit the structure of the Sylvester matrix to increase their efficiency with controlling the size of coefficients well. Although, in this paper, we have primarily focused our attention into reducing the structure of the nested subresultant matrix to "flat" representation, development of more efficient algorithm such as exploiting the structure of the Sylvester matrix would be the next problem. Furthermore, the reduced nested subresultant may involve fractions which may be unusual for subresultants, hence more detailed analysis of computational efficiency including comparison with (ordinary and recursive) subresultants would also be necessary.

We expect that the reduced nested subresultants can be used for including approximate algebraic computation, especially for the square-free decomposition of approximate univariate polynomials with approximate GCD computations based on Singular Value Decomposition (SVD) of subresultant matrices ([12] [13]), which motivates the present work. We can calculate approximate square-free decomposition of the given polynomial $P(x)$ by several methods including calculation of the approximate GCDs of $P(x), \ldots, P^{(n)}(x)$ (by $P^{(n)}(x)$ we denote the n-th derivative of $P(x)$) or those of the recursive PRS for $P(x)$ and $P'(x)$; as for these methods, we have to find the representation of the subresultant matrices for $P(x), \ldots, P^{(n)}(x)$, or that for the recursive PRS for $P(x)$ and

$P'(x)$, respectively. While several algorithms based on different representation of subresultant matrices have been proposed ([14] [15]) for the former approach, we expect that our reduced nested subresultant matrix can be used for the latter approach. To make use of the reduced nested subresultant matrix, we need to reveal the relationship between the structure of the subresultant matrices and their singular values; this is the problem on which we are working now.

References

1. Knuth, D.: The Art of Computer Programming. Third edn. Volume 2: Seminumerical Algorithms. Addison-Wesley (1998)
2. Collins, G.E.: Subresultants and Reduced Polynomial Remainder Sequences. J. ACM **14** (1967) 128–142
3. Brown, W.S., Traub, J.F.: On Euclid's Algorithm and the Theory of Subresultants. J. ACM **18** (1971) 505–514
4. Loos, R.: Generalized polynomial remainder sequences. In Buchberger, B., Collins, G.E., Loos, R., eds.: Computer Algebra: Symbolic and Algebraic Computation. Second edn. Springer-Verlag (1983) 115–137
5. Terui, A.: Subresultants in recursive polynomial remainder sequence. In Ganzha, V., Mayr, E., Vorozhtsov, E., eds.: Proc. The Sixth Computer Algebra in Scientific Computing: CASC 2003, München, Institute für Informatik, Technische Universität München (2003) 363–375
6. Terui, A.: Recursive polynomial remainder sequence and calculation of the number of real zeros of univariate polynomial (in Japanese). In Noro, M., ed.: Computer algebra—algorithms, implementations and applications (Kyoto, 2003). Number 1395 in RIMS Collection of Research Reports. Research Institute for Mathematical Sciences, Kyoto Univ., Kyoto (2004) 97–103
7. von zur Gathen, J., Lücking, T.: Subresultants revisited. Theoret. Comput. Sci. **297** (2003) 199–239 Latin American theoretical informatics (Punta del Este, 2000).
8. Bareiss, E.H.: Sylvester's identity and multistep integer-preserving Gaussian elimination. Math. Comp. **22** (1968) 565–578
9. Golub, G.H., Van Loan, C.F.: Matrix Computations. Third edn. The Johns Hopkins University Press (1996)
10. Ducos, L.: Optimizations of the subresultant algorithm. J. Pure Appl. Algebra **145** (2000) 149–163
11. Lombardi, H., Roy, M.F., El Din, M.S.: New structure theorem for subresultants. J. Symbolic Comput. **29** (2000) 663–689
12. Corless, R.M., Gianni, P.M., Trager, B.M., Watt, S.M.: The singular value decomposition for polynomial systems. In: Proc. ISSAC 1995, ACM (1995) 195–207
13. Emiris, I.Z., Galligo, A., Lombardi, H.: Certified approximate univariate GCDs. J. Pure Appl. Algebra **117/118** (1997) 229–251 Algorithms for algebra (Eindhoven, 1996).
14. Diaz-Toca, G.M., Gonzalez-Vega, L.: Barnett's theorems about the greatest common divisor of several univariate polynomials through Bezout-like matrices. J. Symbolic Comput. **34** (2002) 59–81
15. Rupprecht, D.: An algorithm for computing certified approximate GCD of n univariate polynomials. J. Pure and Applied Algebra **139** (1999) 255–284

Interdependence Between the Laurent-Series and Elliptic Solutions of Nonintegrable Systems

S.Yu. Vernov

Skobeltsyn Institute of Nuclear Physics, Moscow State University,
Vorob'evy Gory, Moscow, 119992, Russia
svernov@theory.sinp.msu.ru

Abstract. The standard methods for the search for the elliptic solutions consist of two independent steps: transformation of a nonlinear polynomial differential equation into a nonlinear algebraic system and the search for solutions of the obtained system. It has been demonstrated by the example of the generalized Hénon–Heiles system that the use of the Laurent-series solutions of the initial differential equation assists to solve the obtained algebraic system and, thereby, simplifies the search for elliptic solutions. This procedure has been automatized with the help of the computer algebra systems Maple and REDUCE. The Laurent-series solutions also assist to solve the inverse problem: to prove the non-existence of elliptic solutions. Using the Hone's method based on the use the Laurent-series solutions and the residue theorem, we have proved that the cubic complex one-dimensional Ginzburg–Landau equation has neither elliptic standing wave nor elliptic travelling wave solutions. To find solutions of the initial differential equation in the form of the Laurent series we use the Painlevé test.

1 Introduction

The investigations of exact special solutions of nonintegrable systems play an important role in the study of nonlinear physical phenomena. There are a few methods to construct solutions in terms of rational, hyperbolic, trigonometric or elliptic functions (all such functions are solutions of the first-order polynomial differential equations) [1–8]. These methods (at least some of them) use information about the dominant behavior of the initial system solutions in the neighbourhood of their singular points, but do not use the Laurent series representations of them. In [9] R. Conte and M. Musette developed a new method, based on the Painlevé analysis and the Laurent-series solutions, as an alternative way to construct elliptic and elementary solutions. In this paper we demonstrate that this method is useful not only as an alternative one but also in combination with traditional methods for construction of special solutions. To show it we consider the generalized Hénon–Heiles system, for which analytic and Laurent-series solutions have been found in [4,7,10].

We also consider the one-dimensional cubic complex Ginzburg–Landau equation (CGLE) [11]. All known single-valued solutions of this equation are elementary functions. In [9] they have been recovered by the Conte–Musette method. The non-existence of elliptic travelling wave solutions of the CGLE has been proved by A.N.W. Hone [12]. This result is based on analysis of the Laurent-series solutions and the residue theorem. In the paper we prove that the CGLE (with special choice of parameters given in [9]) has neither elliptic standing wave nor elliptic travelling wave solutions.

It has been noted by R. Conte and M. Musette in [13] that a computer algebra package is highly recommended for the use of their method. The corresponding package in Maple [19] has been presented in [14]. This package constructs the system of algebraic equations, which corresponds to the given Laurent-series. In this paper we describe a new procedure of this package, which assists to simplify the obtained algebraic system. Also we show how our package assists to prove the non-existence of elliptic solutions.

2 Metnods for Construction of Elliptic Solutions

2.1 The Hénon–Heiles System

To compare the methods of the elliptic solutions construction let us consider the generalized Hénon–Heiles system with an additional non-polynomial term, which is described by the Hamiltonian:

$$H = \frac{1}{2}\Big(x_t^2 + y_t^2 + \lambda_1 x^2 + \lambda_2 y^2\Big) + x^2 y - \frac{C}{3}\,y^3 + \frac{\mu}{2x^2} \tag{1}$$

and the corresponding system of the motion equations:

$$\begin{cases} x_{tt} = & -\lambda_1 x - 2xy + \dfrac{\mu}{x^3}, \\ y_{tt} = & -\lambda_2 y - x^2 + Cy^2, \end{cases} \tag{2}$$

where subscripts denote derivatives: $x_{tt} \equiv \frac{\mathrm{d}^2x}{\mathrm{d}t^2}$ and $y_{tt} \equiv \frac{\mathrm{d}^2y}{\mathrm{d}t^2}$, λ_1, λ_2, μ and C are arbitrary numerical parameters. Note that if $\lambda_2 \neq 0$, then one can put $\lambda_2 = sign(\lambda_2)$ without loss of generality. If $C = 1$, $\lambda_1 = 1$, $\lambda_2 = 1$ and $\mu = 0$, then (2) is the initial Hénon–Heiles system [16].

The function y, solution of system (2), satisfies the following fourth-order polynomial equation:

$$\begin{aligned} y_{tttt} = (2C-8)y_{tt}y - (4\lambda_1+\lambda_2)y_{tt} + 2(C+1)y_t^2 + \\ + \frac{20C}{3}y^3 + (4C\lambda_1 - 6\lambda_2)y^2 - 4\lambda_1\lambda_2 y - 4H. \end{aligned} \tag{3}$$

The energy of the system H is not an arbitrary parameter, but a function of initial data: y_0, y_{0t}, y_{0tt} and y_{0ttt}. The form of this function depends on μ.

2.2 Construction of a Nonlinear Algebraic System

The direct algebraic method is the substitution of the first-order polynomial differential equation, which solutions are elementary or elliptic functions, into the initial differential system to transform it into a nonlinear algebraic system in coefficients of the first-order equation and parameters of the initial system. The obtained system of algebraic equations in principle can be solved by the Gröbner (Groebner) bases method [17], but calculations for a sufficiently complex algebraic system can be very difficult and expensive. The use of the Laurent-series solutions gives additional algebraic equations and allows to simplify calculations. These equations are linear in coefficients of the first-order equation and nonlinear, maybe even nonpolynomial, in parameters of the initial system. Note that one maybe should fix some of these parameters to construct the Laurent-series solutions. Therefore, in contrast to the Gröbner bases method, the additional equations may be not consequences of the initial algebraic system.

Let

$$y(t) = \varrho(t)^2 + P_0, \tag{4}$$

where P_0 is a constant, then eq. (3) is equivalent to

$$\begin{aligned}\varrho_{tttt}\varrho &= 2(C-4)\varrho_{tt}\varrho^3 - 4\varrho_{ttt}\varrho_t - 3\varrho_{tt}^2 + (2P_0(C-4) - 4\lambda_1 - \lambda_2)\varrho_{tt}\varrho + \\ &+ 2(3C-2)\varrho_t^2\varrho^2 + (2CP_0 - 4\lambda_1 - 8P_0 - \lambda_2)\varrho_t^2 + \frac{10}{3}C\varrho^6 + \\ &+ (2C\lambda_1 + 10CP_0 - 3\lambda_2)\varrho^4 + 2(2\lambda_1 CP_0 + 5CP_0^2 - \lambda_1\lambda_2 - \\ &- 3P_0\lambda_2)\varrho^2 + \frac{10}{3}CP_0^3 + 2\lambda_1 CP_0^2 - 3P_0^2\lambda_2 - 2\lambda_1\lambda_2 P_0 - 2H.\end{aligned} \tag{5}$$

We will seek such solutions of (5) that they are the general solution of the following first-order equation

$$\varrho_t^2 = \frac{1}{4}\Big(A_4\varrho^4 + A_3\varrho^3 + A_2\varrho^2 + A_1\varrho + A_0\Big), \tag{6}$$

where A_k are constants to be determined.

Using (6) we transform eq. (5) into the following algebraic system:

$$\left\{\begin{aligned} &(3A_4 + 4)\,(2C - 3A_4) = 0, \\ &A_3(9C - 21A_4 - 16) = 0, \\ &96A_4CP_0 - 240A_4A_2 - 192A_4\lambda_1 - 384A_4P_0 - 48A_4\lambda_2 - \\ &\; - 105A_3^2 + 128A_2C - 192A_2 + 128C\lambda_1 + 640CP_0 - 192\lambda_2 = 0, \\ &40A_3CP_0 - 90A_4A_1 - 65A_3A_2 - 80A_3\lambda_1 - \\ &\; - 160A_3P_0 - 20A_3\lambda_2 + 56CA_1 - 64A_1 = 0, \\ &16A_2CP_0 - 36A_4A_0 - 21A_3A_1 - 8A_2^2 - 32A_2\lambda_1 - 64A_2P_0 - 8\lambda_2A_2 + \\ &\; + 24CA_0 + 64\lambda_1CP_0 + 160CP_0^2 - 16A_0 - 32\lambda_1\lambda_2 - 96P_0\lambda_2 = 0, \\ &10A_3A_0 + (5A_2 + 8CP_0 - 16\lambda_1 - 32P_0 - 4\lambda_2)A_1 = 0 \end{aligned}\right. \tag{7}$$

and the equation for the energy H:

$$H = \frac{1}{384}\Big(96CA_0P_0 - 48A_2A_0 + 384C\lambda_1P_0^2 + 640CP_0^3 - 9A_1^2 - \\ - 192A_0\lambda_1 - 384A_0P_0 - 48A_0\lambda_2 - 384\lambda_1\lambda_2P_0 - 576\lambda_2P_0^2\Big). \tag{8}$$

In [5] it has been proposed to seek solutions as polynomials with three arbitrary coefficients

$$y(t) = P_2\varrho(t)^2 + P_1\varrho(t) + P_0, \tag{9}$$

where $\varrho(t)$ satisfies eq. (6). The function

$$\breve{\varrho}(t) = \frac{1}{\sqrt{P_2}}\left(\varrho(t) - \frac{P_1}{2}\right) \tag{10}$$

satisfies eq. (6) as well, therefore, the same function $y(t)$ corresponds to a two-parameter set of coefficients A_i $(i = 0..4)$ and P_k $(k = 0..2)$, so we always can put $P_2 = 1$ and $P_1 = 0$, in other words use (4) instead of (9) without loss of generality.

System (7) has been solved by REDUCE [18] using the standard function **solve** and the Gröbner bases method [7]. The goal of this paper is to show that the use of the Laurent-series solutions allows us to obtain some solutions of (7) solving only linear systems and nonlinear equations in one variable.

We cannot use this method for arbitrary C, because the Laurent-series solutions are different for different C. So first of all we have to fix the value of C. To fix C we use the condition $A_3 \neq 0$, then from two first equations of system (7) it follows:

$$C = -\frac{4}{3} \quad \text{and} \quad A_4 = -\frac{4}{3} \qquad \text{or} \qquad C = -\frac{16}{5} \quad \text{and} \quad A_4 = -\frac{32}{15}. \tag{11}$$

2.3 Construction of a Linear Algebraic System

Let us choose $C = -4/3$ (one can consider the case $C = -16/5$ similarly). If we consider system (7) separately from the differential equations (6) and (5), from which it has been obtained, then it would be difficult to solve system (7) without the use of the Gröbner bases method. At the same time from equations (6) and (5) we can obtain an additional information, which assists us to solve system (7).

Let us construct the Laurent-series solutions for eq. (5). The method of construction of the Laurent-series solutions for the generalized Hénon–Heiles system has been described in detail in [10]. For eq. (5) with $C = -4/3$ we obtain that solutions have singularities proportional to $1/t$ and the values of resonances are -1 (corresponds to an arbitrary parameter t_0), 1, 4 and 10. The Laurent-series solutions are (we put $t_0 = 0$):

$$\tilde{\rho} = \pm\left(\frac{i\sqrt{3}}{t} + c_0 + \frac{i\sqrt{3}}{24}\left(3\lambda_2 - 2\lambda_1 + 4P_0 + 62c_0^2\right)t + \dots\right), \tag{12}$$

where

$$c_0 = \frac{\pm\sqrt{161700\lambda_1 - 121275\lambda_2 \pm \sqrt{1155(5481\lambda_2^2 - 12768\lambda_1\lambda_2 + 8512\lambda_1^2)}}}{2310}. \qquad (13)$$

Two signs " ± " in (13) are independent. At the same time, functions $\tilde{\rho}$ and $-\tilde{\rho}$ correspond to one and the same function $\tilde{y}$, so there are four different Laurent-series solutions. The coefficients c_3 and c_9 are arbitrary. To find any number of coefficients, we should solve, except linear equations in one variable, only two nonlinear equations in one variable.

Using the algorithm of the construction of elliptic solutions, described in [9], we substitute the obtained Laurent-series solutions in eq. (6). This substitution transforms eq. (6) into a linear and overdetermined system in A_k with coefficients depending on arbitrary parameters.

The obtained system has the triangular form and is linear not only in A_k, but also in H, c_3 and c_9. From the first equation we obtain anew that $A_4 = -4/3$. From the second equation it follows that

$$A_3 = \frac{16}{3}c_0, \qquad \text{and so on:} \qquad (14)$$

$$A_2 = -70c_0^2 - 3\lambda_2 + 2\lambda_1 - 4P_0, \qquad (15)$$

$$A_1 = \left(\frac{40}{3}P_0 - 60\lambda_1 + 50\lambda_2 + 1300c_0^2\right)c_0, \qquad (16)$$

$$\begin{aligned} A_0 = & -40i\sqrt{3}c_3 - \frac{21535}{12}c_0^4 + \left(\frac{565}{6}\lambda_1 - \frac{405}{4}\lambda_2 - \frac{245}{3}P_0\right)c_0^2 + \\ & + \frac{7}{4}\lambda_1\lambda_2 - \frac{21}{16}\lambda_2^2 - \frac{7}{12}\lambda_1^2 + \frac{7}{3}\lambda_1 P_0 - \frac{7}{2}\lambda_2 P_0 - \frac{7}{3}P_0^2. \end{aligned} \qquad (17)$$

From the next equation of the system we obtain c_3 and, finally,

$$\begin{aligned} A_0 = & \frac{15645}{4}c_0^4 + \left(\frac{1545}{4}\lambda_2 - 465P_0 - \frac{1495}{2}\lambda_1\right)c_0^2 + \frac{537}{20}\lambda_1^2 - \\ & - \frac{663}{20}\lambda_1\lambda_2 + \frac{729}{80}\lambda_2^2 + 19\lambda_1 P_0 - \frac{37}{2}\lambda_2 P_0 - \frac{17}{3}P_0^2. \end{aligned} \qquad (18)$$

Substituting the values of A_k, which correspond to one of the possible values of c_0, in system (7), we obtain that it is satisfied for all values of λ_1, λ_2 and P_0, so we do not need to solve the nonlinear equations. Therefore, we settle the nonlinear algebraic system (7) in the case $C = -4/3$, solving only linear equations and nonlinear equation in one variable. We have used the values of only six coefficients of the Laurent series solutions. Note that, for c_0 and $-c_0$ we obtain the same values of A_4, A_2 and A_0, and the opposite values of A_3 and A_1. From (6) it follows that these solutions correspond to $\pm\rho(t)$, and, hence, give the same function $y(t)$. Therefore, two rather than four different elliptic (or degenerated elliptic) solutions of eq. (3) have been found.

To obtain the explicit form of the elliptic function, which satisfies the known first-order ODE, one can use the classical method due to Poincaré, which has been implemented in Maple [19] as the package "algcurves" [20]. The elliptic solutions $y(t)$ are the fourth-order elliptic functions and can be expressed in terms of the Weierstrass elliptic function $\wp(t-t_0)$:

$$y(t-t_0) = \left(\frac{a\wp(t-t_0)+b}{c\wp(t-t_0)+d}\right)^2 + P_0, \tag{19}$$

where constants a, b, c, d and periods of $\wp(t)$ are determined by A_k. The parameters t_0 and P_0, which define the energy of the system, are arbitrary. Solutions of this type exist in both above-mentioned nonintegrable cases: $C = -16/5$ and $C = -4/3$. Full list of solutions is given in [7].

3 Non-existence of Elliptic Solutions

3.1 The Complex Ginzburg–Landau Equation

The one-dimensional cubic complex Ginzburg–Landau equation (CGLE) [11] is one of the most well studied nonlinear equations (see [21] and references therein). It is a generic equation which describes many physical phenomena, such as pattern formation near a supercritical Hopf bifurcation [21] or spatiotemporal intermittency in spatially extended dissipative systems [22]. The CGLE

$$\mathrm{i}A_t + pA_{xx} + q|A|^2A - \mathrm{i}\gamma A = 0, \tag{20}$$

where subscripts denote partial derivatives: $A_t \equiv \frac{\partial A}{\partial t}$, $A_{xx} \equiv \frac{\partial^2 A}{\partial x^2}$, $p \in \mathbb{C}$, $q \in \mathbb{C}$ and $\gamma \in \mathbb{R}$ is not integrable if $pq\gamma \neq 0$. In the case $q/p \in \mathbb{R}$ and $\gamma = 0$ the CGLE is the well-known nonlinear Schrödinger equation. One of the most important directions in the study of the CGLE is the consideration of its travelling wave reduction [9,12,21,23,24]:

$$A(x,t) = \sqrt{M(\xi)}\,\mathrm{e}^{\mathrm{i}(\varphi(\xi)-\omega t)}, \quad \xi = x - ct, \qquad c \in \mathbb{R}, \quad \omega \in \mathbb{R}, \tag{21}$$

which defines the following third-order system

$$\begin{cases} \dfrac{M''}{2M} - \dfrac{M'^2}{4M^2} - \left(\psi - \dfrac{cs_r}{2}\right)^2 - \dfrac{cs_iM'}{2M} + d_rM + g_i = 0, \\ \psi' + \left(\psi - \dfrac{cs_r}{2}\right)\left(\dfrac{M'}{M} - cs_i\right) + d_iM - g_r = 0, \end{cases} \tag{22}$$

where $\psi \equiv \varphi' \equiv \frac{d\varphi}{d\xi}$, $M' \equiv \frac{dM}{d\xi}$, six real parameters d_r, d_i, g_r, g_i, s_r and s_i are given in terms of c, p, q, γ and ω as

$$d_r + \mathrm{i}d_i = \frac{q}{p}, \quad s_r - \mathrm{i}s_i = \frac{1}{p}, \quad g_r + \mathrm{i}g_i = \frac{\gamma + \mathrm{i}\omega}{p} + \frac{1}{2}c^2s_is_r + \frac{\mathrm{i}}{4}c^2s_r^2. \tag{23}$$

Using (22) one can express ψ in terms of M and its derivatives:

$$\psi = \frac{cs_r}{2} + \frac{G' - 2cs_iG}{2M^2(g_r - d_iM)}, \tag{24}$$

where

$$G \equiv \frac{1}{2}MM'' - \frac{1}{4}M'^2 - \frac{cs_i}{2}MM' + d_rM^3 + g_iM^2, \tag{25}$$

and obtain the third order equation in M:

$$(G' - 2cs_iG)^2 - 4GM^2(d_iM - g_r)^2 = 0. \tag{26}$$

To avoid carrying heavy expressions, following [9], we put

$$p = -1 - 3\mathrm{i}, \qquad q = 4 - 3\mathrm{i}. \tag{27}$$

Substituting these values of p and q, we obtain

$$d_r = \frac{1}{2}, \quad d_i = \frac{3}{2}, \qquad s_r = -\frac{1}{10}, \quad s_i = -\frac{3}{10}. \tag{28}$$

Equation (26) in the case $p/q \notin \mathbb{R}$ is nonintegrable, which means that the general solution (which should depend on three arbitrary integration constants) is not known. It has been shown using the Painlevé analysis [24] or topological arguments [23] that single-valued solutions can depend on only one arbitrary parameter. Equation (26) is autonomous, so this parameter is ξ_0: if $M = f(\xi)$ is a solution, then $M = f(\xi - \xi_0)$, where $\xi_0 \in \mathbb{C}$, has to be a solution. All known exact solutions of the CGLE are elementary (rational, trigonometric or hyperbolic) functions. The full list of these solutions is presented in [9,12].

A.N.W. Hone [12] has proved that a necessary condition for eq. (26) to admit elliptic solutions is $c = 0$. In this paper we prove that eq. (26) does not admit elliptic solutions in the case $c = 0$ as well. In other words, neither travelling nor standing wave solutions are elliptic functions. In contrast to [9,12] we consider system (22) instead of eq. (26). Below we show that this choice has some preferences.

3.2 Elliptic Functions

The function $\varsigma(z)$ of the complex variable z is a doubly-periodic function if there exist two numbers ω_1 and ω_2 with $\omega_1/\omega_2 \notin \mathbb{R}$, such that for all $z \in \mathbb{C}$

$$\varsigma(z) = \varsigma(z + \omega_1) = \varsigma(z + \omega_2). \tag{29}$$

By definition a double-periodic meromorphic function is called an elliptic function. These periods define the period parallelograms with vertices z_0, $z_0 + N_1\omega_1$, $z_0 + N_2\omega_2$ and $z_0 + N_1\omega_1 + N_2\omega_2$, where N_1 and N_2 are arbitrary natural numbers and z_0 is an arbitrary complex number. The classical theorems for elliptic functions (see, for example, [25]) prove that

- If an elliptic function has no poles then it is a constant.
- The number of elliptic function poles within any finite period parallelogram is finite.

- The sum of residues within any finite period parallelogram is equal to zero (**the residue theorem**).
- If $\varsigma(z)$ is an elliptic function then any rational function of $\varsigma(z)$ and its derivatives is an elliptic function as well.

From (24) it follows that if M is an elliptic function then ψ has to be an elliptic function. Therefore, if we prove that ψ cannot be an elliptic function, we prove that M cannot be an elliptic function as well. To prove this we construct the Laurent-series solutions for system (22) and apply the residue theorem to the function φ and its degrees.

3.3 Nonexistence of the Standing Wave Elliptic Solutions

It has been proved [12] that if ψ is a constant then M cannot be a nonconstant elliptic function. So, to obtain nontrivial elliptic solutions we have to assume that ψ has poles. We do not restrict ourselves to the case $c = 0$ and prove the non-existence of either travelling or standing wave solutions. It has been noted in [13] that one does not need to transform a system of differential equations into one equation to obtain the Laurent-series solutions.

Using the Ablowitz–Ramani–Segur algorithm of the Painleve test [15] and a computer algebra system (for example, Maple [19] or REDUCE [18]) it is easy to find the Laurent-series solutions of system (22). There exist two different Laurent-series solutions ($\xi_0 = 0$):

$$\begin{aligned}\psi_1 = & -\frac{1}{\xi} + \frac{1}{6}\left(\frac{9}{200}c^2 - \frac{5}{2}g_i - g_r\right)\xi + \frac{1}{40}\left(\frac{3}{100}c^3 - \frac{3}{2}cg_i + \frac{1}{3}cg_r\right)\xi^2 + \\ & + \frac{1}{180}\left(\frac{81}{40000}c^4 - \frac{81}{200}c^2 g_i + \frac{39}{100}c^2 g_r + \frac{61}{4}g_i^2 + 11 g_i g_r + g_r^2\right)\xi^3 + \dots\end{aligned} \tag{30}$$

and

$$\begin{aligned}\psi_2 = & \frac{2}{\xi} - \frac{3c}{20} - \frac{1}{39}\left(\frac{27}{200}c^2 - 10g_i - g_r\right)\xi - \frac{1}{260}\left(\frac{3}{50}c^3 - 3cg_i - \frac{7}{6}cg_r\right)\xi^2 + \\ & - \frac{1}{1521}\left(\frac{3969}{400000}c^4 - \frac{963}{1000}c^2 g_i - \frac{561}{500}c^2 g_r + \frac{122}{5}g_i^2 + 8g_i g_r - \frac{7}{2}g_r^2\right)\xi^3 + \dots\end{aligned} \tag{31}$$

Solutions ψ_1 and ψ_2 correspond to solutions of eq. (26)

$$M_- = -\frac{2}{\xi^2} + \dots \qquad \text{and} \qquad M_+ = \frac{4}{\xi^2} + \dots,$$

which have been found in [9].

The sum of residues of all poles of an elliptic function within some finite period parallelogram is equal to zero. So the elliptic function $\psi(x)$ has to have both ψ_1 and ψ_2 Laurent series expansions. Let the function ψ have N_1 poles with residues, which are equal to 2, within some finite period parallelogram, then in this domain the number of poles, which residues are equal to -1, has to be equal to $2N_1$. So, if the function ψ is elliptic, then within some finite period parallelogram it has N_1 poles with ψ_1 Laurent expansion and $2N_1$ poles with ψ_2 Laurent expansion. If ψ is an elliptic function, then ψ^2 is an elliptic function as well and satisfies the residue theorem (the residues of powers of ψ have been calculated with the help of the procedure *ydegree* from our package of Maple procedures [14], which realizes the Conte–Musette algorithm for construction of single-valued solutions of nonintegrable systems [9]). Residues of ψ_1^2 are equal to zero, whereas residues of ψ_2^2 are $-3c/5$. So we obtain the condition $c = 0$ and prove the absence of the travelling wave solutions. Applying the residue theorem to ψ^3 and ψ^5 and using the condition $c = 0$, we obtain the following system on parameters g_r and g_i:

$$\begin{cases} 5g_i - 6g_r = 0, \\ 1827g_i^2 - 2076g_ig_r - 356g_r^2 = 0. \end{cases} \tag{32}$$

System (32) is satisfied only if

$$g_i = 0 \qquad \text{and} \qquad g_r = 0. \tag{33}$$

In this case the Laurent-series solutions give

$$\psi_1 = -\frac{1}{\xi}, \quad M_1 = -\frac{2}{\xi^2}, \qquad \text{and} \qquad \psi_2 = \frac{2}{\xi}, \quad M_2 = \frac{4}{\xi^2}. \tag{34}$$

The straightforward substitution of these functions in system (22) with $c = 0$, $g_r = 0$ and $g_i = 0$ proves that they are exact solutions. The coefficients of the Laurent-series solutions do not include arbitrary parameters, so the obtained solutions are unique single-valued solutions, and the CGLE has no elliptic solution for these values of parameters as well. Thus, we have proved the non-existence of both travelling and standing wave elliptic solutions. We have proved the non-existence of standing wave solutions for the special choice of parameters s_r, s_i, d_r and d_i. If the initial values of these parameters are not zero then using the scaling transformations, we always can select the values of s_r, s_i, d_r as in (28) without loss of generality, whereas the value of d_i remains arbitrary. In [12] the nonexistence of elliptic travelling wave solutions has been proved for an arbitrary d_i. We prove the nonexistence of elliptic standing wave solutions of the CGLE with an arbitrary d_i in [26]. Note that in general case the system of differential equations can give more information about the existence or the non-existence of elliptic solutions, than the equivalent differential equation in one variable.

4 The Corresponding Computer Algebra Procedures

All calculations have been made with the help of the package of computer algebra procedures, which had been written in Maple and REDUCE. The Maple version of this package has been presented on the International Conference CASC'04 and has been published in the proceedings of this conference [14].

Let us consider this package, which constructs the first order equation in the form (1) with the given (formal) Laurent-series solution:

$$y = \sum_{k=-p}^{N_{max}} c(k) t^k, \qquad (35)$$

where p and N_{max} are some integer numbers. The general form of the first-order polynomial differential equation with solutions (35) is [9,13]

$$F(y_t, y) \equiv \sum_{k=0}^{m} \sum_{j=0}^{j<=(m-k)(p+1)/p} h_{j,k}\, y^j y_t^k = 0, \qquad h_{0,m} = 1. \qquad (36)$$

At singular points y_t^m tends to infinity as $1/t^{m(p+1)}$, so we can present $F(y_t, y)$ as the Laurent series, beginning from this term:

$$F(y_t, y) = \sum_{s=-m(p+1)}^{N_{max}-m(p+1)+p} K_s t^s \qquad (37)$$

and transform (36) into the overdetermined algebraic system: $K_s = 0$ in $h_{i,j}$.

The procedure *equalist*(h, m, p) constructs a list, which corresponds to the first-order equations (36) with unknown coefficients $h_{i,j}$, m is the highest order of derivative in (36). The procedure *quvar*(m, p) calculates the number of unknown coefficients $h_{i,j}$. The procedure *equlaurlist*(h, m, p, ove, c) transforms the first order differential equation into an algebraic system and constructs the first *quvar*(m, p) + *ove* equations of this system.

We can use the fact that $c(-p) \neq 0$ to exclude some of unknowns $h_{i,j}$ from the obtained algebraic system, which is linear in them. We add to our package the procedure *simlequa*$(flist, h, m, p)$, which gives the corresponding $h_{i,j}$. If one does not restrict the general form of (36), one has to use $flist :=$ *equalist*(h, m, p). If one fixes some $h_{i,j}$, then one has to exclude them from $flist$. For example, in Section 2 we construct eq. (36) for $m = 2, p = 1$ and put $h_{2,1} = 0$, $h_{1,1} = 0$ and $h_{0,1} = 0$, so we have used $flist := [[h[0,0], 0, 0], [h[1,0], 1, 0]$, $[h[2,0], 2, 0], [h[3,0], 3, 0], [h[4,0], 4, 0], [h[0,2], 0, 2]]$ or equivalently

$$flist := [[h[0,0],0,0],[h[1,0],1,0],[h[2,0],2,0],[h[3,0],3,0],[h[4,0],4,0]]; \quad (38)$$

The procedure simplequa(flist,h,2,1) gives the list

$$[[-4, 4, 0], [-3, 3, 0], [-2, 2, 0], [-1, 1, 0], [0, 0, 0]]. \qquad (39)$$

This result means that we can exclude $h_{4,0}$ from system (37), using the equation, which corresponds to t^{-4}, exclude $h_{3,0}$, using the equation, which corresponds to t^{-3}, and so on. To calculate residues of products $y(t)^n y'(t)^m$ one can use the procedure monomlaur(c,mon,j,p), with $j = -1$ and $mon = [1, n, m]$, but previously one has to put $\forall k = -m(p+1) - np.. - p - 1 : c(k) = 0$.

The procedure *monomlaur* is presented in [14]. The considering packages of procedures in Maple and REDUCE are available in Internet [27].

5 Conclusion

The Laurent-series solutions are useful to find elliptic or elementary solutions in the analytic form. The method proposed in [9] converts the local information into the global one and can be used not only as an alternative of the standard method, but also as an addition to it, which assists to find solutions of the obtained algebraic system. We have demonstrated that one can find elliptic solutions of the generalized Hénon–Heiles system solving only linear equations and nonlinear equations in one variable, instead of nonlinear system (7). At the same time, to use this method one has to know not only an algebraic system, but also differential equations, from which this system has been obtained.

The Laurent-series solutions are useful not only to find elliptic solutions, but also to prove the non-existence of them. Using the Hone's method based on residue theorem, we have proved the non-existence of both standing and travelling wave elliptic solutions of the CGLE.

Acknowledgements

The author is grateful to R. Conte, V.F. Edneral and A.N.W. Hone for valuable discussions. This work has been supported by Russian Federation President's Grant NSh–1685.2003.2 and by the grant of the scientific Program "Universities of Russia" 02.02.503.

References

1. Weiss, J.: Bäcklund transformation and linearizations of the Hénon–Heiles system. Phys. Lett. A **102** (1984), 329–331; Bäcklund transformation and the Hénon–Heiles system, Phys. Lett. A **105** (1984) 387–389
2. Santos, G.S.: Application of finite expansion in elliptic functions to solve differential equations, J. Phys. Soc. Japan **58** (1989) 4301–4310
3. Conte, R., Musette, M.: Linearity inside nonlinearity: exact solutions to the complex Ginzburg–Landau equation. Phisica D **69** (1993) 1–17
4. Timoshkova, E.I.: A New class of trajectories of motion in the Hénon–Heiles potential field. Astron. Zh. **76** (1999) 470–475 {in Russian}; Astron. Rep. **43** (1999) 406–411, {in English}
5. Fan, E.: An algebraic method for finding a series of exact solutions to integrable and nonintegrable nonlinear evolutions equations. J. Phys. A **36** (2003) 7009–7026
6. Kudryashov, N.A.: Nonlinear differential equations with exact solutions expressed via the Weierstrass function, nlin.CD/0312035
7. Timoshkova, E.I., Vernov, S.Yu.: On two nonintegrable cases of the generalized Hénon–Heiles system with an additional nonpolynomial term. math-ph/0402049, Yadernaya Fizika (Physics of Atomic Nuclei) **68**, No. 11 (2005), in press
8. Kudryashov, N.A.: Simplest equation method to look for exact solutions of nonlinear differential equations, nlin.SI/0406007
9. Musette, M., Conte, R.: Analytic solitary waves of nonintegrable equations. Physica D **181** (2003) 70–79, nlin.PS/0302051

10. Vernov, S.Yu.: Construction of solutions for the generalized Hénon–Heiles system with the help of the Painlevé test. TMF (Theor. Math. Phys.) **135** (2003) 409–419 {in Russian}; 792–801 {in English}, math-ph/0209063
11. Ginzburg, V.L., Landau, L.D.: On the theory of superconductors, Zh. Eksp. Teor. Fiz. (Sov. Phys. JETP) **20** (1950) 1064–1082 {in Russian}, in: Landau, L.D., Collected Papers, Pergamon Press, Oxford (1965) p. 546 {in English}
12. Hone, A.N.W.: Non-existence of elliptic travelling wave solutions of the complex Ginzburg–Landau equation. Physica D **205** (2005) 292–306
13. Conte, R., Musette, M.: Solitary waves of nonlinear nonintegrable equations. nlin.PS/0407026
14. Vernov, S.Yu.: Construction of single-valued solutions for nonintegrable systems with the help of the Painlevé test, in: Proc. Int. Conference "Computer Algebra in Scientific Computing" (CASC'04, St. Petersburg, Russia, 2004), eds. V.G. Ganzha, E.W. Mayr, E.V. Vorozhtsov, Technische Universitat, Munchen, Garching (2004) 457–465, nlin.SI/0407062
15. Ablowitz, M.J., Ramani, A., Segur, H.: A connection between nonlinear evolution equations and ordinary differential equations of P-type. I & II. J. Math. Phys. **21** (1980) 715–721 & 1006–1015
16. Hénon, M., Heiles, C.: The applicability of the third integral of motion: Some numerical experiments. Astronomical J. **69** (1964) 73–79
17. Davenport, J.H., Siret, Y., Tournier E., Calcul Formel, Systemes et Algorithmes de Manipulations Algebriques, Masson, Paris, New York (1987)
18. Hearn, A.C.: REDUCE. User's Manual, Vers. 3.8, `http://www.reduce-algebra.com/documentation.htm` REDUCE. User's and Contributed Packages Manual, Vers. 3.7, CA and Codemist Ltd, Santa Monica, California (1999) `http://www.zib.de/Symbolik/reduce/more/moredocs/reduce.pdf`
19. Heck, A.: Introduction to Maple, 3rd Edition, Springer–Verlag, New York (2003)
20. van Hoeij, M.A.: package "algcurves", Maple V and Maple 6, `http://www.math.fsu.edu/~hoeij/maple.html`
21. Aranson, I., Kramer, L.: The world of the complex Ginzburg–Landau equation. Rev. Mod. Phys. **74** (2002) 99–143, cond-mat/0106115
22. van Hecke, M., Storm, C., van Saarlos, W.: Sources, sinks and wavenumber selection in coupled CGL equations and experimental implications for counter-propagating wave systems. Phisica D **133** (1999) 1–47, Patt-sol/9902005
23. van Saarloos, W., Hohenberg, P.C.: Fronts, pulses, sources and sinks in generalized complex Ginzburg–Landau equations. Phisica D **56** (1992) 303–367, Erratum **69** (1993) 209
24. Cariello, F., Tabor, M.: Painlevé expansions for nonintegrable evolution equations. Phisica D **39** (1989) 77–94
25. Erdélyi, A., et al. (eds.), Higher Transcendental Functions (based, in part, on notes left by H. Bateman), Vol. 3, MC Graw-Hill Book Company, New York (1955)
26. Vernov, S.Yu.: On elliptic solutions of the cubic complex one-dimensional Ginzburg–Landau equation, nlin.PS/0503009
27. Vernov, S.Yu.: `http://theory.sinp.msu.ru/~svernov/programs`

Solving Linear Differential Problems with Parameters

Volker Weispfenning

University of Passau, Germany
weispfen@uni-passau.de
http://www.fmi.uni-passau.de/algebra/staff/weispfen.php3

Abstract. We present algorithms for parametric problems in differential algebra that can be formulated in a suitable first-order language L. The atomic L-formulas are linear ODEs of arbitrary order with parametric coefficients of arbitrary degrees. Using rather weak axioms on differential fields or differential algebras that are realized in natural function domains, we establish explicit quantifier elimination algorithms for L providing also parametric sample solutions for purely existential problems. These sample solutions are "generic" solutions of univariate parametric linear ODEs that can be realized by concrete functions in the natural function domains mentioned above. We establish upper complexity bounds for the elimination algorithms that are elementary recursive for formulas of bounded quantifier alternation, in particular doubly exponential for existential formulas. Our results are in contrast to Seidenberg's model theoretic elimination theory for non-linear problems that is non elementary recursive, requires very strong axioms that are not realizable in natural function domains, and does not provide sample solutions.

1 Introduction

In the 1950s A. Seidenberg developped an ingenious algebraic and algorithmic elimination theory for parametric ODEs (and also PDEs) [1]. This theory gave rise to the model theoretic concept of a differentially closed field as an (almost) perfect analogue for an algebraically closed field [2–4]. Later L. Blum found simple axioms for differentially closed fields [5,6]. Recently the joint algorithmic outcome of all these theoretical results has been implemented by A. Dolzmann and T. Sturm in the REDLOG package of the computer algebra system REDUCE [7]. This approach has, however, two significant drawbacks: Firstly, it is valid only in the "artificial algebraic paradise" of differentially closed fields, that can never be realized by a natural analytic function domain, due to the strength of its axioms. Secondly, the elimination algorithm uses an enormous number of case distinctions and iterated formation of disjunctive and conjunctive normal forms of quantifier-free formulas that may result in a combinatorial explosion. As a result the elimination algorithm is non elementary recursive in the worst case; this may even happen for input formulas of bounded quantifier alternation.

The present paper is an attempt to avoid both drawbacks by restricting attention to a fragment of the first-order language of differential fields, where all quantified variables occur only linearly, whereas parameters may occur in arbitrary degrees. By analysing L.Blums's axioms for this situation, we find that they can be weakened in such a way that they are realizable in natural analytic functions domains. In particular these domains need neither be algebraically closed fields nor real closed fields.

Thus the corresponding elimination theory becomes analytically meaningful. By a considerable modification of Seidenberg's method we develop an explicit quantifier elimination algorithms for the restriced language L and the weaker axioms providing also parametric sample solutions for purely existential linear problems. These sample solutions are "generic" solutions of univariate parametric linear ODEs that can be realized by concrete functions in the the natural function domains mentioned above. We establish upper complexity bounds for the elimination algorithms that are elementary recursive for formulas of bounded quantifier alternation, in particular doubly exponential for existential formulas and thus considerably better than those for Seidenberg's method.

In fact we also present a variant of all these results for the even more restricted case, where parametric variables range only over differntial constants. In this case the axioms can be further weakened, so that they are e. g. satisfied by by a differential algebra of complex polynomials in x and functions of type $e^{\lambda x}$ for complex λ or a differential algebra of real polynomials in x and functions of type $e^{\lambda x}, \sin(\lambda x), \cos(\lambda x)$ for real λ.

As a byproduct we get decion algorithms for parameter-free formulas in these languages with the same upper complexity bounds.

Some simple examples illustrate the range and applicability of these theoretical results. An implementation in the REDLOG package of REDUCE is planned.

2 Basic Concepts

A differential ring (differential field) F is an integral domain (a field) F extending the field $\mathbb{Q}$ of rational numbers together with a formal derivation, i. e.unary operation $' : F \longrightarrow F$ satisfying

$$(a+b)' = a' + b', \quad (a \cdot b)' = a \cdot b' + a' \cdot b$$

An element a of a differential ring F is a *constant* if $a' = 0$. The set K of differential constants of a differential ring F forms a subring of F, the *constant subring* of F. If F is a field, so is K. We call a differential ring F a *differential algebra*, if K is a field [8].

The first-order theory DF of differential fields has the natural language $L = \{0, 1, +, -, \cdot, '\}$. In this language it is easy to formulate the axioms for DF by the axioms for fields of characteristic zero together with the sum and product rule above for the derivation. Similarly, one can formulate axioms DA for differential algebras in L. For practical applicability we may also allow in L constants for some or all elements of F provided these objects are in a fixed subalgebra of F and can be handled algorithmically, i.e. for every variable-free term one should be able to decide, whether it represents zero or not.

The terms of this language may all be written as (ordinary) differential polynomials in some variables $y_1, \dots, y_n$, in other words as polynomials in $y_1, \dots, y_n$, and their iterated derivatives $y_i^{(j)}$ with integer coefficients. The *(formal) order* $\mathrm{ord}_y(t)$ of a variable y in a term t is the highest superscript j such that $y^{(j)}$ occurs in t. Let y be of order k in t. Then the *initial* $\mathrm{init}_y(t)$ of y in t is the coefficient of the highest power of $y^{(k)}$ in t.

In order to get a nice elimination theory in DF, i. e. quantifier elimination in the language L one needs rather strong additional axioms of the following kind [5,6]: First one requires the fields to be algebraically closed; second one requires for every pair of terms t, s in L and every variable y such that $\mathrm{ord}_y(t) > \mathrm{ord}_y(s)$ the axiom

$$(\mathrm{init}_y(t) \neq 0 \wedge \mathrm{init}_y(s) \neq 0) \implies \exists y(t = 0 \;\wedge\; s \neq 0)$$

By adding these axioms one passes from the theory DF of differential fields to the theory CDF of *differentially closed differential fields.* Notice that these axioms imply that both the field F and its constant field K have to be algebraically closed.

This stronger theory has very nice algebraic and model theoretic properties, in particular it admits an algorithmic quantifier elimination and a decision procedure [1,2,5,6,3,4]. On the other hand differentially closed differential fields can never be realized as natural fields of functions. So these fields are a kind of "algebraic paradise" far removed from the "real world" of analysis.

Here we consider therefore the much weaker theories $LCDF$ of linear-closed differential fields and $WLCDA$ of weakly linear-closed differential algebras that do in fact admit many models that are differential function rings; thus these theories is really close to actual analytic problems.

For these weaker theories we provide explicit axioms, efficient algorithmic quantifier elimination and decision procedures for suitable classes of L-formulas and compute upper complexity bounds for these procedures. In order to describe $LCDF$ we first specify a restricted class of L-formulas: Suppose we have partitioned the set of all variables in L into two disjoint infinite blocks, the set LV of *linear variables* and the set PV of *parametric variables.* Then we define *linear terms* as differential polynomials with integer coefficients in which linear variables and their higher derivatives occur only linearly, whereas parametric variables and their iterated derivatives may occur in arbitrary degrees. Thus linear terms can be written as linear polynomials in the linear variables and their iterated derivatives with coefficients that are in turn arbitrary differential polynomials with integer coefficients in the parametric variables only. We call a linear term t *weakly parametric* if all coefficients of linear variables and their iterated derivatives in t are variable-free terms representing fixed elements of F. So parametric variables occur at most in the absolute term of t wrt. the linear variables and their iterated derivatives. A linear term t is *univariate* if it contains at most one linear variable together with its iterated derivatives. If this linear variable is y, then we say t is *univariate in y.* If in addition t has no absolute term wrt. y, then we call t *homogeneous univariate in y.*

Atomic linear formulas are then equations between linear terms. Atomic linear formulas are *weakly parametric* if the terms in them are weakly parametric. Arbitrary *linear formulas* are obtained from atomic linear formulas by propositional connectives "and", "or", "not" and quantifications $\exists x$ or $\forall x$ over linear variables $x \in LV$. A linear formula is *weakly parametric* if all its atomic subformulas are weakly parametric. A linear term or a linear formula is *purely parametric* if it contains no linear variable. We allow purely parametric terms also to be written in non-expanded form.

We deal only with problems that can be formulated by linear formulas. Due to this restriction we can now fomulate much weaker axioms than those for CDF in order to obtain quantifier elimination and decision procedures for linear formulas:

We require for every natural number n and every $n+1$-tuple of linear terms $t, s_1, \ldots, s_n$ in L that are univariate in a linear variable y such that $ord(y,t) > ord(y, s_i)$ for all $1 \leq i \leq n$, the axiom

$$(\text{init}_y(t) \neq 0 \wedge \bigwedge_{i=1}^{n} \text{init}_y(s_i) \neq 0) \implies \exists y(t = 0 \ \wedge \ s_1 \neq 0 \wedge \ldots \wedge s_n \neq 0)$$

We call the theory with these new axioms the theory $LCDF$ of *linear-closed differential fields.* Notice that for CDF there was no necessity to consider more than one inequality, since one may form products of differential terms, which is in general impossible for linear differential terms.

An even weaker theory $WLCDA$ of *weakly linear closed differential algebras* is obtained, when takes the axioms of DA together with the axioms above restricted to the case where all coefficients of y and its iterated derivatives in t and in all s_i are required to be differential constants, i. e. elements of K.

3 Examples of Linear Closed Differential Fields

Which differential fields satisfy the axioms of LCDF and which differential algebras satisfy the axioms of WLCDA? In order to appraoch this question, it is useful to formulate potentially stronger sets of axioms that are closer to the standard theory of ODEs:

Call a differential field F *plentiful,* if the following two conditions are satisfied:

1. Every linear univariate ODE of positive order with coefficients in F has a solution in F.
2. The set of solutions of a homogeneous univariate linear ODE with coefficients in F of positive order n forms a vector space of dimension n over the field K of differential constants of F.

Call a differential algebra F *weakly plentiful,* if the corresponding two conditions are satisfied for linear univariate ODEs with coefficients in K.

By the standard theory of linear ODEs any differential field F that satisfies condition 2 and is closed under integration is in fact plentiful.

It is now easy to see, that every plentiful differential field F is indeed linear closed: Given a linear univariate ODE $t = 0$ of positive order n, and finitely many further linear univariate ODEs $s_i = 0$ of order smaller than n, all with coefficients in F, we may conclude that the set of solutions of $t = 0$ in F forms a non-empty affine space A over K of dimension n; similarly the solution set S_i of each $s_i = 0$ in F forms an affine space over K of dimension smaller than n. Hence the union of all S_i can never cover all of A. This shows that there is a solution of $t = 0$ in F which is not a solution of every equation $s_i = 0$.

A similar argument shows that every weakly plentiful differential algebra is weakly linear-closed.

A related argument also shows that the axioms for linear closed differential fields can be weakened as follows: It suffices to require for every natural number n and every $n+1$-tuple of linear terms $t, s_1, \dots, s_n$ in L that are homogeneous univariate in a linear variable y such that $ord(y,t) > ord(y,s_i)$ for all $1 \le i \le n$, the axiom

$$(\mathrm{init}_y(t) \neq 0 \wedge \bigwedge_{i=1}^{n} \mathrm{init}_y(s_i) \neq 0) \implies \exists y(t = 0 \ \wedge \ s_1 \neq 0 \wedge \dots \wedge s_n \neq 0)$$

In addition one must require for every linear term u univariate in a linear variable y with $ord(y,u) > 0$ the axiom

$$(\mathrm{init}_y(t) \neq 0 \implies \exists y(t = 0)$$

All examples listed below will be plentiful differential fields or weakly plentiful differential algebras:

Fix a non-empty open connected region $G \subseteq \mathbb{C}$. Then the differential field of all meromorphic functions defined on G is plentiful. This applies in particular to regions $G \subseteq \mathbb{R}$ and also to the differential field of all real-valued meromorphic functions defined on G. Hence all these fields are linear-closed.

The differential algebra A generated as a ring by $\mathbb{C}$ together with the identity function and all the exponential functions $e^{\lambda x}$ for arbitrary $\lambda \in \mathbb{C}$ is a weakly plentiful differential algebra. Its elements are complex polynomials in x and all $e^{\lambda x}$. Notice that in the representation of elements of A powers $(e^{\lambda x})^k$ of $e^{\lambda x}$ are superfluous, since they can be rewritten as $e^{k\lambda x}$.

Similarly, the differential algebra A generated as a ring by $\mathbb{C}$ together with the identity function all the exponential functions $e^{\lambda x}$, and the trigonometric functions $\cos(\mu x)$, $\sin(\nu x)$ for $\lambda, \mu, \nu \in \mathbb{R}$ is a weakly plentiful differential algebra. Its elements are real polynomials in $x, e^{\lambda x}, \cos(\mu x), \ \sin(\nu x)$. Again powers of $e^{\lambda x}, \cos(\mu x), \ \sin(\nu x)$ can be avoided in the representation of elements of A by the identities

$$(e^{\lambda x})^k = e^{k\lambda x},$$

$$\sin(\mu x)\sin(\nu x) = 2^{-1}(\cos((\mu - \nu)x) - \cos((\mu + \nu)x))$$

$$\cos(\mu x)\cos(\nu x) = 2^{-1}(\cos((\mu - \nu)x) + \cos((\mu + \nu)x)).$$

Hence all these algebras are weakly linear-closed.

4 Reduction and Normal Forms

First we describe a reduction process for linear atomic formulas with respect to a specified linear variable y that is analogous to Gauss elimination and in fact may be considered as a parametric version of Ritt reduction [8,1].

Notice that every linear term t can be written wrt. a specified linear variable y in the form

$$t = a_k y^{(k)} + \ldots + a_1 y' + a_0 y + a_{-1},$$

where $\mathrm{init}_y(t) := a_k, \ldots, a_1$ are purely parametric differential polynomials, and $\mathrm{abs}_y(t) := a_{-1}$ is a linear term not containing the variable y. If, moreover, t is a weakly parametric linear term, then $a_k, \ldots, a_1$ are variable-free terms representing fixed integers.

We call this form of t a *normal form* of t wrt. y. The derivative t' of a linear term t as above is then

$$t' = a_k y^{(k+1)} + (a_k' + a_{(k-1)})y^{(k)} + \ldots + (a_2' + a_1)y^{(2)} + (a_1' + a_0)y' + a_0' y + a_{-1}'$$

which is again in normal form wrt. y with $\mathrm{init}_y(t') = a_k = \mathrm{init}_y(t)$. If in addition t is weakly parametric, then $a_k' = \ldots = a_0' = 0$, and so

$$t' = a_k y^{(k+1)} + a_{(k-1)} y^{(k)} + \ldots + a_1 y^{(2)} + a_0 y' + a_{-1}'$$

Iterating this observation for higher derivatives, we find that for $m \in \mathbb{N}$,

$$t^{(m)} = a_k y^{(k+m)} + t^*,$$

where t^* is a linear term in normal form wrt. y of order $\mathrm{ord}_y(t^*) < k + m$; in particular $\mathrm{init}_y(t^{(m)}) = a_k = \mathrm{init}_y(t)$ and $\mathrm{ord}_y(t^{(m)}) = m + k$.

The *reductum* $\mathrm{red}_y(t) = a_{k-1} y^{(k-1)} + \ldots + a_0$ of t wrt. y is simply obtained from t by deleting the highest order monomial of t in normal form wrt. y. *Iterated reducta* are defined by

$$\mathrm{red}_y(t)^0 := t, \quad \mathrm{red}_y(t)^{i+1} = \mathrm{red}_y(\mathrm{red}_y(t)^i)$$

for $i < k$. So $\mathrm{init}_y(\mathrm{red}_y(t)^i) = a_{k-i}$.

Let s, t be two linear terms both in normal form with respect to y and assume $k := \mathrm{ord}_y(s) \leq \mathrm{ord}_y(t) =: m$. Then we define the reduction t_1 of t wrt. y and s, $t \underset{y,s}{\longrightarrow} t_1$ by setting $t_1 := \mathrm{init}_y(s)t - \mathrm{init}_y(t)s^{(m-k)}$.

Since $s^{(m-k)}$ is of the form $s^{(m-k)} = \mathrm{init}_y(s)y^{(m)} + s^*$, where s^* is a linear term with $\mathit{ord}(y, s^*) < m$, it follows that $\mathit{ord}(y, t^*) < \mathit{ord}(y, t) = m$. Moreover in every differential algebra, we have

$$(\mathrm{init}_y(s) \neq 0 \wedge s = 0) \implies (t = 0 \iff t_1 = 0)$$

Iterating this type of reduction for fixed y and s, we obtain reduction chains leading from t in say r steps to t_r, $t \overset{r}{\underset{y,s}{\longrightarrow}} t_r$. If this reduction chain is of maximal length $r \leq m - k + 1$, then we call t_r *the normal form* of t wrt. y and s. If the number $r \geq 0$ is irrelevant we simply write $t \xrightarrow[y,s]{*} t_r$. By induction on the number r of reduction steps the equation above holds as well for reduction chains:

$$(\mathrm{init}_y(s) \neq 0 \wedge s = 0) \implies (t = 0 \iff t_r = 0)$$

Notice that the reductum of t as defined above has nothing to do with reductions of t.

5 Generic Solutions

Let $t = a_k y^{(k)} + \ldots + a_1 y' + a_0 y + a_{-1}$ be a linear term in normal form wrt. the variable y. For later use we define $\mathrm{guard}_{y,t} := (a_0 = \ldots = a_k = 0)$. Notice that this is a quantifier-free linear formula containing no linear variables.

Next we introduce a new formal expression $\mathrm{gen}_{y,t}$, called the *generic solution* of $t = 0$ wrt. y.

The semantics of $\mathrm{gen}_{y,t}$ is as follows: Suppose we have fixed all values of all parametric variables in a differential algebra A such that $\mathrm{init}_y(t) \neq 0$ holds in A, and we have specified a finite set S of linear terms such that for these specified values of the parameters $\mathrm{ord}_y(s) < \mathrm{ord}_y(t)$ for all $s \in S$. Then $\mathrm{gen}_{y,t}$ is an object in A satisfying

$$t = 0 \ \wedge \bigwedge_{s \in S} s \neq 0$$

when substituted for y in these linear terms.

The existence of such an object is guaranteed, if A is a linear closed differential field. Alternatively, if all coefficients of $y, \ldots, y^{(k)}$ in t and in all $s \in S$ have values in the constant field K, then the existence of such an object is already guaranteed, if only A is a weakly linear closed differential algebra.

Notice that of course this object depends on the values in A of the parametric variables occurring in t and in all $s \in S$. Moreover it depends on the set S itself. The latter dependence can be removed by taking $\mathrm{gen}_{y,t}$ as an object in a suitable elementary extension A^* of A. If we insist on $\mathrm{gen}_{y,t}$ being an object in A, then we specify the dependence of this object on S by saying that *$gen_{y,t}$ is a generic solution of $t = 0$ wrt. y and S.*

Anyway, the only fact we need of $\mathrm{gen}_{y,t}$ is its correct semantic behaviour, when formally substituted for y in some linear atomic formula φ. So our next goal is to define such a modified substitution in such a way that the expression $\mathrm{gen}_{y,t}$ does in fact not occur in the resulting linear formula.

Let φ be of the form $v = 0$, where v is a linear term in normal form wrt. y. Then we define the modified substitution of $\mathrm{gen}_{y,t}$ for y in $v = 0$, notation $(v = 0)[\mathrm{gen}_{y,t}//y]$, as follows: Let v^* be the normal form of v upon reduction of v wrt. y and t. Then the implication

$$(\mathrm{init}_y(t) \neq 0 \wedge t = 0) \implies (v = 0 \Leftrightarrow v^* = 0)$$

is valid in every differential algebra.

Let now v^* be in normal form wrt. y,

$$v_i = b_j y^{(j)} + \ldots + b_0 y + b_{-1}$$

with $j < \mathrm{ord}_y(t)$. Then we let $(v = 0)[\mathrm{gen}_{y,t}//y]$ be the linear formula $b_j = \ldots = b_0 = b_{-1} = 0$.

Notice that this definition conforms with the semantic properties of $\mathrm{gen}_{y,t}$ specified above, i.e. if in some differential algebra A, where $\mathrm{gen}_{y,t}$ exists and $\mathrm{init}_y(t) \neq 0$ holds for specific values of the parametric variables, then $v = 0$ holds in A at point $y = \mathrm{gen}_{y,t}$ iff $(v = 0)[\mathrm{gen}_{y,t}//y]$ holds in A.

We extend the concept of modified substitution of $\mathrm{gen}_{y,t}$ for y to arbitrary quantifier-free linear formulas ψ in the natural way: So $\psi[\mathrm{gen}_{y,t}//y]$ is obtained from ψ by performing the modified substitution $\varphi \mapsto \varphi[\mathrm{gen}_{y,t}//y]$ in every atomic subformula of ψ. Then this modified substitution is again semantically correct: So if in some differential algebra A, where $\mathrm{gen}_{y,t}$ exists and $\mathrm{init}_y(t) \neq 0$ holds for specific values of the parametric variables, then ψ holds in A at point $y = \mathrm{gen}_{y,t}$ iff $\psi[\mathrm{gen}_{y,t}//y]$ holds in A. Moreover by the remark above we see that the number of atomic subformulas does not increase in the passage from ψ to $\psi[\mathrm{gen}_{y,t}//y]$, provided t and ψ are both weakly parametric.

One important observation that follows from the semantics of generic solutions is the following

Lemma 1. *Let t be a linear term in normal form wrt. the linear variable y and assume $ord_y(t) \geq 0$. Let U be a finite set of linear terms that do not contain the variable y, and let S be another set of linear terms in normal form wrt. y. Then in every linear closed differential field the following holds:*

$$(init_y(t) \neq 0 \ \wedge t = 0 \ \wedge \ \bigwedge_{u \in U} u = 0 \ \wedge \ \bigwedge_{s \in S} s \neq 0) \implies$$

$$(t[gen_{y,t}//y] = 0 \ \wedge \ \bigwedge_{u \in U} u[gen_{y,t}//y] = 0 \ \wedge \ \bigwedge_{s \in S} s[gen_{y,t}//y] \neq 0)$$

Proof. Assume the hypothesis. Since all $u \in U$ do not involve y, we get $u[\mathrm{gen}_{y,t}//y] = u$. By the hypothesis we have for every normal form s^* of $s \in S$ that $s^* \neq 0$, and so the fact that at least one coefficient of s^* wrt. y is non-zero. So by the semantics of generic solutions, we have that $s[\mathrm{gen}_{y,t}//y] = s^*[\mathrm{gen}_{y,t}//y] \neq 0$. □

6 Quantifier Elimination for a Single Quantifier

Using the expressions for generic solutions we can now perform quantifier elimination in the theory LCDF and WLCDA, respectively, for linear formulas $\exists x(\varphi)$ involving only a single existential quantifier.

To begin with we let y be a linear variable and denote by T_y (T_y^+) the set of all linear terms that are in normal form with respect to y (and have $\mathrm{ord}_y(t) \geq 0$).

Next we associate with every non-empty finite subset T of T_y a finite set Γ_T of purely parametric formulas that are conjunctions of equations and disequations, and a map $\Gamma_T \longrightarrow T_y^+$, $\gamma \mapsto t_\gamma$, such that the following properties hold in every linear closed differential field, and under the hypothesis that all parametric variables have derivative zero also in all weakly linear closed differential algebras:

1. $(\bigwedge_{u\in T} \mathrm{guard}_{y,u} \wedge \mathrm{abs}_y(u) = 0) \ \vee (\bigvee_{\gamma\in\Gamma_T} \gamma)$
2. For every $\gamma \in \Gamma_T$, $\quad (\gamma \wedge \bigwedge_{u\in T} u = 0) \implies (\mathrm{init}_y(t_\gamma) \neq 0 \wedge t_\gamma = 0 \ \wedge \bigwedge_{u\in T} u[\mathrm{gen}_{y,t_\gamma}] = 0)$.

The definition of Γ_T and the map $\gamma \mapsto t_\gamma$ is by recursion on the natural number $d := \sum_{u\in T}(\mathrm{ord}_y(u) + 1)$.

If $d = 0$ then no $u \in T$ involves y and we put $\Gamma_T = \emptyset$. Note that empty disjunctions are false and empty conjunctions are true by definition; so in this case the properties above are satisfied.

If $d > 0$ and there is at most one $u \in T$ which involves y, then we put

$$\Gamma_T := \{\mathrm{init}_y(u) \neq 0\} \ \cup \ \{(\mathrm{init}_y(u) = 0 \ \wedge \ \gamma \mid \gamma \in (\Gamma_{(T\setminus\{u\}) \ \cup \ \{\mathrm{red}_{y(u)}\}},$$

and put

$$t_{\mathrm{init}_y(u)\neq 0} := u, \ t_{\mathrm{init}_y(u)=0\wedge\gamma} := t_\gamma.$$

Then the first property is obvious and the second follows directly from the induction assumption.

If $d > 0$ and T contains at least two terms s, t with $\mathrm{ord}_y(s), \mathrm{ord}_y(t) \leq 0$, and say $\mathrm{ord}_y(s) \leq \mathrm{ord}_y(t)$, then we let $s^* := \mathrm{red}_y(s)$, and let t^* be the normal form of t wrt. y and s. Then we put

$$\Gamma_T := \{(\mathrm{init}_y(s) \neq 0 \ \wedge \ \gamma)|\gamma \in \Gamma_{(T\setminus\{t\}) \ \cup \ \{t^*\}}\} \ \cup$$

$$\{(\mathrm{init}_y(s) = 0 \ \wedge \ \delta)|\delta \in \Gamma_{(T\setminus\{s\}) \ \cup \ \{s^*\}}\}.$$

Moreover we put

$$t_{(\mathrm{init}_y(s)\neq 0 \ \wedge \ \gamma)} := t_{\gamma)}, \ \ t_{(\mathrm{init}_y(s):=0 \ \wedge \ \delta)} := t_\delta.$$

Then again the first property is obvious and the second follows directly from the induction assumption applied to $\Gamma_{(T\setminus\{t\}) \ \cup \ \{t^*\}}$ and to $\Gamma_{(T\setminus\{s\}) \ \cup \ \{s^*\}}$.

Finally we put $T^\sim := \{t_\gamma|\gamma \in \Gamma_T\}$, and $T^* := \bigcup_{\emptyset\neq U\subseteq T} U^\sim$.

Notice that if all terms $u \in T$ are weakly parametric, then $T^\sim$ is a singleton set, since all case distinctions on initials are superfluous.

Then we have:

Theorem 1. *Let φ be a linear formula consisting of an $\wedge - \vee$-combination of equations $t = 0$ and disequations $s \neq 0$ for linear terms t, s. Let T be the finite set of all terms $t \in T_y^+$ occuring in some equation $t = 0$ in φ. Let y be a linear variable. Let $q \in \mathbb{N}$ be bigger than all the numbers $\mathrm{ord}_y(s)$, such that s occurs in* *a disequation $s \neq 0$ in φ, and let v be the linear term $y^{(q)}$. Let T^* be the set of linear terms constructed as above from T.*

Then in every linear closed differential field, the formula $\exists y(\varphi)$ is equivalent to

$$\varphi_1 := \quad \varphi[gen_{q,y}v//y]) \ \vee \ \bigvee_{t \in T^*} (init_y(t) \neq 0 \ \wedge \ \varphi[gen_{y,t}//y])$$

Next let $z_1, \dots, z_p$ be all parametric variables occurring in φ. Then in every weakly linearly closed differential algebra, the following formula holds:

$$(\bigwedge_{j=1}^{p} z'_j = 0) \ \longrightarrow \ (\exists y(\varphi) \ \Leftrightarrow \ (\varphi[gen_{q,y}v//y] \vee \bigvee_{t \in T^*} (init_y(t) \neq 0 \wedge \varphi[gen_{y,t}//y])$$

Proof. We prove the first part; the second is analogous. Let F be a linear closed differential field, We begin by the converse direction of the equivalence:

If for fixed values of the variables except y,

$$\varphi[\mathrm{gen}_{q,y}v//y]) \ \vee \ \bigvee_{t \in T^*} (\mathrm{init}_y(t) \neq 0 \ \wedge \ \varphi[\mathrm{gen}_{y,t}//y])$$

holds in F, then we have two cases: If $\varphi[\mathrm{gen}_{y,v}//y])$, then since $\mathrm{init}_y(v) = 1 \neq 0$, we get by the semantics of the modified substitution of generic solutions that φ holds in F at point $y = \mathrm{gen}_{y,v}$, and so $\exists y(\varphi)$ holds in F.

Otherwise there is some $t \in T^*$ such that $(\mathrm{init}_y(t) \neq 0 \ \wedge \ \varphi[\mathrm{gen}_{y,t}//y])$ holds in F. Then again by the semantics of the modified substitution of generic solutions we get that φ holds in F at point $y = \mathrm{gen}_{y,t}$, and so $\exists y(\varphi)$ holds in F.

Next we prove the other direction: Suppose that for fixed values of all variables except y, $\exists y(\varphi)$ holds in F. Let $c \in F$ be such that φ holds in F at point $y = c$. We may assume that φ has been put into disjunctive normal form, say $\bigvee_{i=1}^{m} \varphi_i$, where each φ_i is a conjunction of equations $u = 0$ for $u \in U$ and of disequations $s \neq 0$ for $s \in S$.

Then for some $1 \leq i \leq m$, φ_i holds in F at point $y = c$. Notice that $U \subseteq T$, and so $U \ \subseteq T^*$.

By the first property of Γ_U,

$(\bigwedge_{u \in U} \mathrm{guard}_{y,u} \wedge \mathrm{abs}_y(u) = 0) \ \vee (\bigvee_{\gamma \in \Gamma_U} \gamma$

holds in F. Again we have two cases:

If $(\bigwedge_{u \in U} \mathrm{guard}_{y,u} \wedge \mathrm{abs}_y(u) = 0)$ holds in F, then $u[\mathrm{gen}_{y,v}//y] = \mathrm{abs}_y(u) = 0$, and for all $s \in S$, $s[\mathrm{gen}_{y,v}//y] \neq 0$. Hnece $\varphi_i[\mathrm{gen}_{y,v}//y]$ holds in F, and so $\varphi[\mathrm{gen}_{y,v}//y]$ holds in F.

Otherwise there is some $\gamma \in \Gamma_U$ such that γ holds in F. Together with $\bigwedge_{u \in U} u = 0$, this implies by the second property of Γ_U, $(\mathrm{init}_y(t_\gamma) \neq 0 \wedge \bigwedge_{u \in U} u[\mathrm{gen}_{y,t_\gamma}] = 0)$.

Concerning the disequations, let for $s \in S$ $\ s^*$ be the normal form of s wrt y and t_γ. Then by the second property of Γ_U, $t_\gamma = 0$ at point $y = c$. Consequently, $s^* = s \neq 0$ in f at point $y = c$. Hence by the semantics of generic solutions, $s[\mathrm{gen}_{y,t_\gamma}//y] \neq 0$. So $\varphi_i[\mathrm{gen}_{y,t_\gamma}//y]$ holds in F, and hence $\varphi[\mathrm{gen}_{y,t_\gamma}//y]$ holds in F.

This completes the proof. $\square$

Remark. The proof actually show more: If φ is already a conjunction of equations and disequations, then the role of the set T^* in the theorem can be replaced by the smaller set $T^\sim$. If in addition φ is weakly parametric, then $T^\sim$ is a singleton set, and so the disjunction over $T^\sim$ has only one disjunct.

7 Quantifier Elimination - The General Case

We begin with quantifier elimination for *existential linear formulas,* i. e. formulas of the form

$$\exists y_1 \ldots \exists y_n(\varphi),$$

where $y_1, \ldots y_n$ are linear variables, φ is a quantifier-free linear formula with parametric variables $z_1, \ldots, z_p$. In the following we assume without restriction that φ is actually a $\wedge$-$\vee$-combination of equations and inequations with right hand side 0. In the special case, where φ is in addition a conjunction of equations and disequations of this form, such a formula is called a *primitive linear formula.*

From the main theorem of the previous section we obtain by induction on n the following result:

Theorem 2. *For every existential linear formula $\psi := \exists y_1 \ldots \exists y_n(\varphi)$ as above one can construct finite sets $T_1, \ldots, T_n$ of linear terms, such that the terms in T_i contain besides parametric variables at most the linear variables $y_1, \ldots, y_i$, and such that ψ is equivalent in LCDF to the disjunction*

$$\psi_n := \bigvee_{t_1 \in T_1} \cdots \bigvee_{t_n \in T_n} (init_{y_n}(t_n) \neq 0 \wedge \ldots \wedge init_{y_1}(t_1) \neq 0) \wedge$$

$$\varphi[gen_{y_n,t_n}//y_n] \ldots [gen_{y_1,t_1}//y_1]$$

Here the iterated modified substitution is performed sequentially from left to right. Moreover the sets T_i can be replaced by considerably smaller set $T_i^\sim$ in case ψ is a primitive linear formula. If in addtion φ is weakly parametric, then all $T_i^\sim$ are in fact singleton sets.

A corresponding result holds for WLCDA under the hypothesis that the derivatives of all parametric variables in ψ have value zero.

Proof. As noted above we assume that φ is a $\wedge$-$\vee$-combination of equations and inequations with right hand side 0. Then the proof is by induction on n. The case $n = 1$ is identical to the previous theorem with the only modification that the conjunct $1 \neq 0$ is added in the first disjunct.

Next assume the theorem for fixed n and let $\psi := \exists y_1 \ldots \exists y_n \exists y_{n+1}(\varphi)$. Let $\psi' := \exists y_1 \ldots \exists y_n(\varphi)$. Then by induction assumption ψ' is equivalent in LCDF to a disjunction of the form

$$\bigvee_{t_1 \in T_1} \cdots \bigvee_{t_n \in T_n} (\mathrm{init}_{y_n}(t_n) \neq 0 \wedge \ldots \wedge \mathrm{init}_{y_1}(t_1) \neq 0) \wedge$$

$$\varphi[\mathrm{gen}_{y_n,t_n}//y_n] \ldots [\mathrm{gen}_{y_1,t_1}//y_1]$$

So by permuting the existential $\exists y_{n+1}$ quantifier with the disjunction, ψ is equivalent in LCDF to

$$\bigvee_{t_1 \in T_1} \cdots \bigvee_{t_n \in T_n} (\mathrm{init}_{y_n}(t_n) \neq 0 \wedge \ldots \wedge \mathrm{init}_{y_1}(t_1) \neq 0) \wedge$$

$$\exists y_{n+1}(\varphi[\mathrm{gen}_{y_n,t_n}//y_n] \ldots [\mathrm{gen}_{y_1,t_1}//y_1])$$

Next we apply the case $n = 1$ to each disjunct and find T_{n+1} such that the above is equivalent in LCDF to

$$\bigvee_{t_1 \in T_1} \cdots \bigvee_{t_n \in T_n} \bigvee_{t_{n+1} \in T_{n+1}} (\mathrm{init}_{y_{n+1}}(t_{n+1}) \neq 0 \wedge \mathrm{init}_{y_n}(t_n) \neq 0 \wedge \ldots \wedge \mathrm{init}_{y_1}(t_1) \neq 0) \wedge$$

$$\varphi[\mathrm{gen}_{y_{n+1},t_{n+1}}//y_n][\mathrm{gen}_{y_n,t_n}//y_n] \ldots [\mathrm{gen}_{y_1,t_1}//y_1]).$$

This proves the induction step. □

Notice that this proof together with the proof of the previous theorem actually shows that we have here a case of *quantifier elimination with answers:* In fact one can easily modify the output of the quantifier elimination algorithm above so that it outputs the finite list of pairs

$$((\mathrm{init}_{y_n}(t_n) \neq 0 \wedge \ldots \wedge \mathrm{init}_{y_1}(t_1) \neq 0), (y_n = \mathrm{gen}_{y_n,t_n}, \ldots, y_1 = \mathrm{gen}_{y_1,t_1})$$

consisting of guards and answers in the sense of A. Dolzmann and T. Sturm. In particular the disjunction over all guards holds in LCDF, and the conjunction of each guard together with the input formula yields the φ evaluated at the point specified in the second entry.

In other words we have a finite choice of guards and corresponding iterated generic solutions of parametric univariate differential equations such that whenever φ has a solution $(y_1, \ldots, y_n) \in F^n$ for a specific values of the parametric variables in a linear closed differential field F, then we can express at least one such solution by a nesting of some iterated generic solutions. The choice of this nesting is decided by the corresponding guard.

Similar statements hold in WCDA under the additional hypothesis that all parametric variables have derivative zero.

Corollary 1. *For every universal linear formula $\psi := \forall y_1 \ldots \forall y_n(\varphi)$ as above one can construct finite sets $T_1, \ldots, T_n$ of linear terms, such that the terms in T_i contain besides parametric variables at most the linear variables $y_1, \ldots, y_i$, and such that ψ is equivalent in LCDF to the conjunction*

$$\psi_n := \bigwedge_{t_1 \in T_1} \cdots \bigwedge_{t_n \in T_n} (\mathit{init}_{y_n}(t_n) \neq 0 \wedge \ldots \wedge \mathit{init}_{y_1}(t_1) \neq 0) \longrightarrow$$

$$\varphi[\mathit{gen}_{y_n,t_n}//y_n] \ldots [\mathit{gen}_{y_1,t_1}//y_1]$$

Here again the iterated modified substitution is performed sequentially from left to right. Moreover the sets T_i can be replaced by considerably smaller set $T_i^{\sim}$ in *case in case φ is a disjunction of equations and disequations. If in addtion φ is weakly parametric, then all $T_i^{\sim}$ are in fact singleton sets. A corresponding result holds for WLCDA under the hypothesis that the derivatives of all parametric variables in ψ have value zero.*

Proof. . Apply the previous theorem to the input formula $\exists y_1 \ldots \exists y_n(\neg\varphi)$, that is equivalent to $\neg\psi$, and negate the result. □

We say that LCDF admits *effective linear quantifier elimination* if there is an algorithm assigning to every linear formula φ a quantifier free linear formula φ' such that the equivalence $\varphi \Leftrightarrow \varphi'$ holds in LCDF. We say WLCDA admits *weak effective linear quantifier elimination* if there is an algorithm assigning to every linear formula φ a quantifier free linear formula φ' both with parametric variables among $z_1, \ldots, z_m$ such that the formula

$$(\bigwedge_{i=1}^{m} z_i' = 0) \implies (\varphi \Leftrightarrow \varphi')$$

holds in WLCDA.

Then we have the following important result as consequence of the theorem and its corollary:

Theorem 3. *LCDF admits effective linear quantifier elimination and WLCDA admits weak effective linear quantifier elimination.*

Proof. Every linear formula φ can be equivalently rewritten in prenex normal form φ_1. Then an easy induction on the number of quantifier blocks in φ_1 shows the result. □

.

We call a linear formula φ a *linear sentence* if every occurence of a variable y in φ is within the scope of a corresponding quantifier $\exists y$ or $\forall y$. We say that LCDF admits *an effective linear decision procedure* if there is an algorithm that takes linear sentences φ as input and outputs "true" respectively "false" if φ respectively $\neg\varphi$ holds in every linear closed differential field. We say WLCDA admits *a weak effective linear decision procedure* if there is an algorithm that takes linear sentences φ whre all constants are differential constants as input and outputs "true" respectively "false" if φ respectively $\neg\varphi$ holds in every weakly linear closed differential algebra.

Corollary 2. *LCDF admits an effective linear decision procedure and WLCDA admits a weak effective linear decision procedure. Moreover all linear closed differential fields satisfy the same linear sentences and all weakly linear closed differential algebras satisfy the same linear sentences, where all constants have derivative zero.*

Proof. Apply first quantifier elimination to the input sentence φ and then decide all atomic subfomulas of the output formula φ' in the fixed field of constants. □

8 Complexity Bounds

In order to get upper bound on the complexity of the quantifier elimination procedures described above in Theorems 2, 3, 5 we need to estimate

1. the change of the orders and of the lengths of linear terms wrt. the linear variables involved in the various algorithms.
2. The number of disjuncts occuring in Theorems 2 and 3.

Obviously the passage from a linear term t to an iterated reductum $red^i(t)$ does neither increase $\mathrm{ord}_z(t)$ nor $\mathrm{length}(t)$ for an arbitrary linear variable z.

For a single reduction $t \underset{y,s}{\longrightarrow} t_1 = \mathrm{init}_y(s)t - \mathrm{init}_y(t)s^{(m-k)}$ we get $\mathrm{ord}_z(t_1) \leq \max(\mathrm{ord}_z(t), \mathrm{ord}_z(s) + \mathrm{ord}_y(t) - \mathrm{ord}_y(s)$. Since $\mathrm{ord}_y(t_1) < \mathrm{ord}_y(t)$, the same bound applies to an iterated reduction: Let $t \overset{*}{\underset{y,s}{\longrightarrow}} t^*$. Then

$$\mathrm{ord}_z(t^*) \leq \max(\mathrm{ord}_z(t), \mathrm{ord}_z(s) + \mathrm{ord}_y(t) - \mathrm{ord}_y(s)).$$

In particular we have

$$\mathrm{ord}_z(t^*) \leq \max(\mathrm{ord}_z(t), \mathrm{ord}_z(s) + \mathrm{ord}_y(t)).$$

Concerning length, a one step reduction of linear terms requires in the coefficients the addition of two products of former coefficients; so we get $\mathrm{length}(t_1) \leq 2(\mathrm{length}(t) + \mathrm{length}(s)) + c$ for some constant integer c. We use this observation in order to compute an upper bound $\mathrm{length}(T^*)$ on $\mathrm{length}(t)$ for all $t \in T^*$ from a corresponding bound $\mathrm{length}(T)$ on on $\mathrm{length}()u$ for all $u \in T$. If $\mathrm{ord}_y(()\varphi) = -1$, then $T^* = \emptyset$, and so we may put $\mathrm{length}(T^*) = 0$. If $\mathrm{ord}_y(()\varphi) \geq 0$, then we may pass to a new linear formula φ_1 determining the same T^* with $\mathrm{ord}_y(()\varphi_1) < \mathrm{ord}_y(()\varphi)$ by one step reductions of linear terms with an additional one step reduction of the original divisor by a resulting term in the worst case. So the length of terms in φ_1 is bounded by $\mathrm{length}(T_1) := 7\mathrm{length}(T)$. By recursion on $\mathrm{ord}_y(()\varphi)$ this yields:

$$\mathrm{length}(T^*) = 7^{\mathrm{ord}_y(()\varphi)}\mathrm{length}(T)$$

By definition the modified substitution of a generic solution of $t = 0$ for some term $t \in T^*$ into some equationψ of the form $s = 0$ or some disequation ψ of the form $s \neq 0$ occuring in φ involves an iterated reduction of s by t, where the number of reduction steps is bounded by $\mathrm{ord}_y(()\varphi)$. Thus again by the argument above the length of the resulting linear formula $\varphi[\mathrm{gen}_{y,t}//y]$ is bounded by:

$$\mathrm{length}(\psi[\mathrm{gen}_{y,t}//y]) \leq 3^{\mathrm{ord}_y(\varphi)}(\mathrm{length}(t) + \mathrm{length}(s)) \leq$$

$$3^{\mathrm{ord}_y(\varphi)}(\mathrm{length}(T^*) + \mathrm{length}(s)) \leq 3^{\mathrm{ord}_y(\varphi)}(7^{\mathrm{ord}_y(\varphi)}\mathrm{length}(T) + \mathrm{length}(s)) \leq$$

$$3^{\mathrm{ord}_y(\varphi)}(7^{\mathrm{ord}_y(\varphi)}\mathrm{length}(T) + \mathrm{length}(s)) \leq 21^{\mathrm{ord}_y(\varphi)}(\mathrm{length}(T))$$

So in total we get

$$\text{length}(\varphi[\text{gen}_{y,t}//y]) \le 21^{\text{ord}_y(\varphi)}\text{length}(\varphi)$$

If φ is a linear formula and y is a linear variable, then we let $\text{ord}_y(\varphi)$ be the maximum of all values $\text{ord}_y(t)$, where t is a linear term occuring in φ. Using the bound above we can now study the increase in order upon modified substitution of a generic solution in a quantifier-free linear formula φ : Recall that the modified substitution is carried out in every atomic subformula of φ. There it involves the formation of the normal form of some term v in φ under reduction wrt. y and t, where t is another term of order bounded by $\text{ord}_y(t) \le \text{ord}_y(\varphi)$. Hence we have

$$\text{ord}_z(\varphi[\text{gen}_{y,t}//y]) \le \text{ord}_z(\varphi) + \text{ord}_y(\varphi).$$

Let $\text{ord}(\varphi)$ be the maximum of all $\text{ord}_z(\varphi)$, where z ranges over all linear variables occuring in φ. The we can summarize the last result in the inequality

$$\text{ord}(\varphi_1) \le 2\text{ord}(\varphi).$$

Next we want to compute an upper bound on the number $D := |T^*| + 1$ of disjuncts occuring in the output formula of Theorem 2. Recall that by definition $|T^*| \le \sum_{U\subseteq T} |U^\sim|$ in the general case. Furthermore $|U^\sim| \le |\Gamma_U|$, and $|\Gamma_U|$ satisfies the following recursive inequalities in $d_U := \sum_{u\in U}(\text{ord}_y(u) + 1)$:

$|\Gamma_U| = 0$ if $d_U = 0$

$|\Gamma_U| \le |\Gamma_{U_1}| + |\Gamma_{U_2}|$, if $d > 0$. Here the induction parameters d_{U_i} for T_i satisfy the inequalities $d_{U_i} \le d_U - 1$.

Hence an induction on d shows that $|\Gamma_U| < 2^d_U$. Consequently we have $D \le |T^*| + 1 \le 1 + \sum_{U\subseteq T} |\Gamma_U| < 1 + \sum_{U\subseteq T} 2^{d_U} \le 1 + \sum_{U\subseteq T} 2^{|U|(\text{ord}_y(\varphi)+1)} \le 2^{|T|}2^{|T|(\text{ord}_y(\varphi)+1)} = 2^{|T|(\text{ord}_y(\varphi)+2)}$.

In the special case, where φ is a conjunction of equations and disequations we can by the remark in section 6 replace the role of T^* by $T^\sim$. So the corresponding bound reads:

$D \le |T^\sim| + 1 \le 1 + |\Gamma_T| \le 2^{d_T} \le 2^{|T|(\text{ord}_y(\varphi)+1)}$.

We denote by $\text{eq}(\varphi)$ the number of equations in φ, and by $\text{eql}(\varphi)$ the number of equations in φ that are not purely parametric, i. e. contain at least one linear variable. Similarly $\text{at}(\varphi)$ is the number of equations in φ, and $\text{atl}(\varphi)$ is the number of equations in φ that are not purely parametric, i. e. contain at least one linear variable.

Then the comparison of these numbers for the input formula φ and each disjunct δ of the output formula φ_1 yields:

$$\text{eql}(\delta) \le \text{eql}(\varphi), \quad \text{eq}(\delta) \le \text{ord}_y(\varphi)\text{eq}(\varphi)$$

and

$$\text{atl}(\delta) \le \text{atl}(\varphi), \quad \text{at}(\delta) \le \text{ord}_y(\varphi)\text{at}(\varphi).$$

Next we analyze the complexity of the algorithm in Theorem 3 by induction on the number n of existential quantifiers in the input formula ψ.

For $n = 1$ the bounds obtained above for Theorem 2 yield:

$$\mathrm{ord}(\psi_1) \leq 2\mathrm{ord}(\psi).$$

Next we compute an upper bound on the number $D(\psi) := |T_1|$ of disjuncts occuring in the output formula $\psi_1$of Theorem 3 in this case, and get:

$$D(\psi_1) \leq 2^{\mathrm{eql}(\psi)(\mathrm{ord}(\psi)+2)}.$$

Finally we obtain for each disjunct δ of the output formula ψ_1 :

$$\mathrm{eql}(\delta) \leq \mathrm{eql}(\psi), \quad \mathrm{eq}(\delta) \leq \mathrm{ord}_y(\psi)\mathrm{eq}(\psi)$$

$$\mathrm{atl}(\delta) \leq \mathrm{atl}(\psi), \quad \mathrm{at}(\delta) \leq \mathrm{ord}_y(\psi)\mathrm{at}(\psi)$$

$$\mathrm{length}(\delta) \leq 2^{5\mathrm{ord}_y(()\psi)}\mathrm{length}(\psi).$$

For the induction step $n \mapsto (n+1)$: we obtain:

$$\mathrm{ord}(\psi_{n+1}) \leq 2\mathrm{ord}(\psi_n)$$

$$D(\psi_{n+1}) \leq D(\psi_n) \cdot 2^{\mathrm{eql}(\psi_n)(\mathrm{ord}(\psi_n)+2)}$$

Furthermore for each disjunct δ of the output formula ψ_{n+1} :

$$\mathrm{eql}(\delta) \leq \mathrm{eql}(\psi_n), \quad \mathrm{eq}(\delta) \leq \mathrm{ord}_y(\psi_n)\mathrm{eq}(\psi_n)$$

$$\mathrm{atl}(\delta) \leq \mathrm{atl}(\psi_n), \quad \mathrm{at}(\delta) \leq \mathrm{ord}_y(\psi_n)\mathrm{at}(\psi_n)$$

$$\mathrm{length}(\delta) \leq 2^{5\mathrm{ord}_y(()\psi_n)}\mathrm{length}(\delta_n).$$

Here δ_n is the longest disjunct in ψ_n.

Using these recursive inequalities, we get the following explicit bounds in Theorem 3:

$$\mathrm{ord}(\psi_n) \leq (n+1)\mathrm{ord}(\psi).$$

$$D(\psi_n) \leq 2^{\mathrm{eql}(\psi)((n+1)\mathrm{ord}(\psi)+2)}.$$

Here the first inequality requires a more detailed analysis of orders of terms in ψ; a straightforward recursion would yield only $\mathrm{ord}(\psi_n) \leq 2^n\mathrm{ord}(\psi)$. Notice that for weakly parametric primitive ψ we have $D(\psi_n) = 1$. For each disjunct δ_n of the output formula ψ_n :

$$\mathrm{eql}(\delta) \leq \mathrm{eql}(\psi).$$

$$\mathrm{atl}(\delta) \leq \mathrm{atl}(\psi).$$

$$\mathrm{eq}(\delta) \leq n!\,\mathrm{ord}(\psi)^n\mathrm{eq}(\psi)$$

$$\mathrm{at}(\delta) \leq n!\,\mathrm{ord}(\psi)^n\mathrm{at}(\psi)$$

$$\mathrm{length}(\delta) \leq 2^{5n!\,\mathrm{ord}(\psi)}\mathrm{length}(\psi).$$

In particular we get the following bounds on the output formula ψ_n under the hypothesis that $\mathrm{atl}(\psi) \geq 1$ and $\mathrm{ord}(\psi) \geq 1$:

$$\mathrm{atl}(\psi_n) \leq 2^{\mathrm{atl}(\psi)(n+3)\mathrm{ord}(\psi)}$$

$$\mathrm{at}(\psi_n) \leq 2^{\mathrm{atl}(\psi)(n+3)\log(n)\mathrm{ord}(\psi)}\mathrm{at}(\psi)$$

$$\mathrm{length}(\psi_n) \leq 2^{(n+1)!\,\mathrm{ord}(\psi)\mathrm{eql}(\psi)}\mathrm{length}(\psi)$$

So roughly speaking, we have the following upper complexity bounds concerning linear quantifier elimination in LCDF and weakly linear quantifier elimination in WLCDA for existential linear formulas with n quantifiers:

Theorem 4. *The order ord(ψ_n) grows linearly in n and in ord(ψ). The number atl(ψ_n) grows doubly exponential in n, and singly exponential in ord(ψ), and in atl(ψ). Similarly, the number at(ψ_n) grows doubly exponential in n, singly exponential in ord(ψ), and in atl(ψ) and in addition linearly in at(ψ). length(psi$_n$) grows doubly exponential in n, singly exponential in ord(ψ), and in eql(ψ) and in addition linearly in length(ψ).*

Notice that all the operations required during quantifier elimination and for deciding atomic linear sentences can be performed in a time bound that is polynomial in the size of the object. So we may conclude:

Corollary 3. *Linear quantifier elimination for existential linear formulas ψ in LCDF can be performed in time doubly exponential in the number of quantifiers of ψ singly exponential in ord(ψ), and in atl(ψ) and linear in length(ψ). Similarly bounds holds for a linear decision procedure for existential linear entences in LCDF. Analogous results hold for the corresponding weak procedures in WLCDA.*

The same bounds apply analogously to (weakly) linear quantifier elimination for universal linear formulas in Corollary 4.

Finally we compute upper complexity bounds for *full linear quantifier elimination.* We consider a prenex linear input formula ψ with b alternating blocks of quantifiers, where each block has at most n quantifiers. Then we get the following recursive upper bounds for the result $\psi_{n,b}$ of the quantifier elimination in Theorem 5 in terms of the recursion in b :

$$\mathrm{ord}(\psi_{n,b+1}) \leq (n+1)\mathrm{ord}(\psi_{n,b}).$$

$$\mathrm{atl}(\psi_{n,b+1}) \leq 2^{\mathrm{atl}(\psi_{n,b})(n+3)\mathrm{ord}(\psi_{n,b})}.$$

$$\mathrm{at}(\psi_{n,b+1}) \leq 2^{\mathrm{atl}(\psi_{n,b})(n+3)\log(n)\mathrm{ord}(\psi_{n,b})}\mathrm{at}(\psi_{n,b}).$$

$$\mathrm{length}(\psi_{n,b+1}) \leq 2^{\mathrm{atl}(\psi_{n,b})(n+1)!\,\mathrm{ord}(\psi_{n,b})}\mathrm{length}(\psi_{n,b}).$$

Using these recursive inequalities, we get the following explicit bounds in Theorem 5:

$$\mathrm{ord}(\psi_{n,b}) \leq (bn+1)\mathrm{ord}(\psi).$$

$$\mathrm{atl}(\psi_{n,b}) \leq \exp(b, \mathrm{atl}(\psi)((n+3)\mathrm{ord}(\psi))^b).$$

$$\mathrm{at}(\psi_{n,b}) \leq \exp(b, \mathrm{atl}(\psi)(n+3)^{3b}\mathrm{ord}(\psi)^b)\mathrm{at}(\psi)$$

$$\mathrm{length}(\psi_{n,b}) \leq \exp(b, \mathrm{atl}(\psi)2^{(n+1)\log(n+1)+2b}\mathrm{ord}(\psi)^b)\mathrm{length}(\psi).$$

Here the function $\exp(b, a)$ is b-times iterated exponentiation with base 2 and argument a. The recursive definition is $\exp(1, a) := 2^a$, $\exp(b+1, a) := 2^{exp(b,a)}$.

So roughly speaking, we have the following upper complexity bounds concerning full linear quantifier elimination in LCDF and weakly linear quantifier elimination in WLCDA for prenex linear formulas with b alternating blocks of at most n quantifiers in each block:

Theorem 5. *Effective quantifier elimination in LCDF applies to a prenex linear input formula ψ with b alternating blocks of at most n quantifiers in each block yields a quantifier-free linear formula $\psi_{n,b}$ with the following features: The order ord($\psi_{n,b}$) grows linearly in bn and in ord(ψ). The number atl($\psi_{n,b}$) grows b-times exponentially in n, ord(ψ), and in atl(ψ). The number at($\psi_{n,b}$) grows b-times exponentially in n, ord(ψ), and in atl(ψ), and in addition linearly in at(ψ). length($\psi_{n,b}$) grows grows b-times exponentially in n, ord(ψ), and in atl(ψ), and in addition linearly in length$(()\psi)$.*

Corollary 4. *Linear quantifier elimination for prenex linear formulas ψ in LCDF with b alternating blocks of at most n quantifiers in each block can be performed within a time bound that grows b-times exponentially in n, ord(ψ), and in atl(ψ), and in addition linearly in length(ψ). Similarly bounds holds for a linear decision procedure for prenex linear entences in LCDF. Analogous results hold for the corresponding weak procedures in WLCDA.*

Thus for a bounded number b of quantifier blocks all these procedures are elementary recursive, while for unbounded b they are in the fourth Grzegorcyk-class (compare [9,10]).

9 Examples

Example 1 Let φ be the linear formula

$$y_2' - y_2 - 12y_1 = 0 \ \wedge\ 3y_2 + y_1 - y_1' = 0 \ \wedge\ y_1' - 7y_1 \neq 0$$

Notice that this formula has no parametric variable. Let A be a weakly linear closed differential algebra. Elimination of y_2 via reduction yields for y_2 the generic solution $z_2 := \mathrm{gen}_{y_2, 3y_2 - y_1' + y_1}$ and for y_1 the linear formula

$$y_1'' - 2y_1' - 35y_1 = 0 \ \wedge\ y_1' - 7y_1 \neq 0$$

Since the order of the disequation is lower than that of the equation, this has a solution in A and we may take for y_1 the generic solution $z_1 := \mathrm{gen}_{y_1, y_1'' - 2y_1' - 35y_1}$. In particular $\exists y_1 \exists y_2(\varphi)$ holds in A and a solution is obtained in the form

$$z_1 := \mathrm{gen}_{y_1, y_1'' - 2y_1' - 35y_1}, \quad z_2 := \mathrm{gen}_{y_2, 3y_2 - z_1' + z_1}$$

Interpreted in the standard weakly linear closed differential algebra A the equation $y_1''-2y_1'-35y_1$ has as fundamental system $\{e^{-5t}, e^{7t}\}$. The given disequation $y_1' - 7y_1 \neq 0$ excludes the second but admits the first solution as a generic one. Substitution in $z_2 := \mathrm{gen}_{y_2, 3y_2 - z_1' + z_1}$ yields $3z_2 = 6e^{-5t}$ and thus $z_2 = 2e^{-5t}$. Thus our solution is as possible in purely algebraic terms.

Example 2. This is a weakly parametric variant of example 1, where the zero on the right hand side of the second equation has beeen replaced by a parameter a. Then the elimination of y_2 and y_2' yields:

$$y_1'' - 2y_1' - 35y_1 = a' - a \ \wedge \ y_1' - 7y_1 \neq 0$$

So if a is a differential constant then $a' = 0$ and $z_1 := a/35$ is a generic solution satisfying this formula, whenever $a \neq 0$. With $z_2 := \mathrm{gen}_{y_2, 3y_2 - z_1' + z_1} = a/3$ we have a complete solution. When we regard a as a linear variable, then $a' \neq 0$ is possible. Taking e. g. $a = e^{\lambda x}$ in A, the generic solution z_1 will look differently for different values of λ which determine the resonance behaviour.

Example 3. Passing to a fully parametric version of Example 1,

$$y_1' = a_1 y_1 + a_2 y_2 + c, \quad y_2' = b_1 y_1 + b_2 y_2 + d, y_1 \neq 0$$

we see that the generic solution of the resulting second order ODE in y_1 will look quite differently in A depending on the eigenvalues of the matrix and inhomogeneous parts. Here we have assumed that parameters have differential constants as values. In the general case, where these values may not be differential constants one has to consider the situation in a linear closed differential field f such as real or complex meromorphic functions. In this case our generic solutions may not even have a closed form in F.

10 Conclusions

We have designed, verified and analyzed algorithms for parametric problems in differential algebra that have been formulated in a suitable first-order language L. The atomic L-formulas were linear ODEs of arbitrary order with parametric coefficients of arbitrary degrees. Using rather weak axioms on differential fields or differential algebras that are realized in natural real or complex analytic function domains, we have established explicit quantifier elimination and decision algorithms for L. For purely existential multivariate problems the elimination algorithms also yields parametric sample solutions for all existentially quantified variables. These sample solutions are "generic" solutions of univariate parametric linear ODEs that can be realized by concrete functions in natural analytic function domains. Thus we have shown that the well-known theory of non-parametric linear ODEs can be extended to a fully parametric situation by replacing concrete solution systems of linear ODEs by"generic" solutions of univariate parametric linear ODEs. Some easy examples have illustrated our approach. We have found upper complexity bounds for the elimination and decision algorithms that are elementary recursive for formulas of bounded quantifier alternation, in particular doubly exponential for existential formulas. Our results are in contrast to Seidenberg's model theoretic elimination theory for non-linear problems that requires very strong axioms that are not realizable in natural function domains, and does not provide sample solutions.

We conjecture that our upper bounds are essentially tight in the worst case. An implementation of the algorithms in the REDLOG package of REDUCE is planned; it will be based on the optimized implementation of Seidenberg's elimination procedure in [7].

References

1. Seidenberg, A.: An elimination theory for differential algebra. Univ. California Publ. Math (N.S.) **3** (1956) 31–66
2. Robinson, A.: Complete theories. North-Holland, Amsterdam (1956)
3. Robinson, A.: Relative model-completeness and the elimination of quantifiers. In Kreisler, H.J., Körner, S., Luxemburg, W.A., Young, A.D., eds.: Abraham Robinson. Selected Papers. Volume 1. Yale University Press, New Haven and London (1979) 146–159 Originally in Dialectica 12 (1958) 394–407. M.R. 21 #1265.
4. Robinson, A.: On the concept of a differentially closed field. In Kreisler, H.J., Körner, S., Luxemburg, W.A., Young, A.D., eds.: Abraham Robinson. Selected Papers. Volume 1. Yale University Press, New Haven and London (1979) 440–455 Originally in Bull. Res. Council Israel 8F (1959), 113–128. M.R. 23 #2323.
5. Blum, L.: Generalized Algebraic Structures: A Model Theoretical Approach. Ph.D. thesis, MIT, Cambridge, MA (1968)
6. Blum, L.: Differentially closed fields: A model theoretic tour. In Bass, H., Cassidy, P.J., Kovacic, J., eds.: Contributions to Algebra. A Collection of Papers Dedicated to Ellis Kolchin. Academic Press, New York (1977) 37–61
7. Dolzmann, A., Sturm, T.: Generalized constraint solving over differential algebras. In Ganzha, V., Mayr, E., Vorozhtsov, E., eds.: Proceedings CASC 2004, TUM (2004)
8. Ritt, J.: Differential Algebra. AMS, New York (1950)
9. Schnorr, C.P.: Rekursive Funktionen und ihre Komplexität. Teubner (1974)
10. Sturm, T., Weispfenning, V.: Quantifier elimination in term algebras. In: Computer Algebra in Scientific Computation - CASC 2002, TU München (2002) 285–30

Approximate Solution of the Dirichlet Problem for Elliptic PDE and Its Error Estimate

Serguey Zemskov

University of Rostock
zemskov@informatik.uni-rostock.de

Abstract. The proposed in [7] uniform error estimate allows to control the accuracy of the symbolic approximate solution of the Dirichlet problem for elliptic PDE in the whole domain of the problem considered. The present paper demonstrates the techniques of finding such an approximate solution with *Mathematica* and the use of the uniform error estimate for a concrete example.

1 Introduction

The modern computer systems for symbolic calculations can noticeably help the researcher in solving differential equations. In the present paper we are going to show how the computer algebra system *Mathematica* [1] can be used in finding a symbolic approximate solution of the Dirichlet problem for elliptic partial differential equation (PDE) and its error estimate.

For the differential equation which has no exact solution (or such a solution can not be found at present), it is often possible to develop methods for finding analytical approximate solution (see, e.g., [2], [3]).

The idea of the error estimate method we use was originally proposed for ordinary differential equations (ODEs) in [4]. The variants of realization of this idea for linear ODEs and non-linear ODEs can be found, for example, in [5] and [6] respectively. Both for the case of ODEs and one of PDEs considered in this paper, it is one of important elements of investigation to guarantee the desired accuracy of the solution found. On the successful resolution of this problem depends whether the approximate solution could have a practical application.

2 The Uniform Error Estimate of the Approximate Solution of the Dirichlet Problem for Elliptic PDE

Let us consider the following Dirichlet problem in the domain $\Omega \subset \mathbb{R}^2$ with doubly continuously differentiable bound Γ.

$$L(U) = (\Delta - a(x,y))\,(U) = f(x,y),\ (x,y) \in \Omega \ , \tag{1}$$

$$U = 0, (x,y) \in \Gamma \ . \tag{2}$$

We suppose here that $a, f \in C^1(\Omega), C(\Omega \bigcup \Gamma)$. We denote by U_L the unknown exact solution of (1), (2) and by U_L^* the approximate one we look for.

We consider also the auxiliary Dirichlet problem

$$\Delta_\kappa(U) = (\Delta - \kappa)\,(U) = f(x,y),\ (x,y) \in \Omega,\ \kappa > 0\ , \tag{3}$$

$$U = 0,\ (x,y) \in \Gamma\ . \tag{4}$$

Let $G_{\Delta_\kappa;\Omega}(x,y;\xi,\eta)$ be the Green's function of (3), (4).

Theorem 1. *If the following inequality holds true*

$$h = \sqrt{\iint\limits_{\Omega} \left(\iint\limits_{\Omega} |(a(\xi,\eta) - \kappa) G_{\Delta_\kappa;\Omega}(\xi,\eta;u,v)|^2 d\xi\, d\eta \right) du\, dv} < 1$$

then there exists the uniform estimate

$$|U_L - U_L^*| \le \frac{||G_{\Delta_\kappa;\Omega}(x,y;.,.)||_2}{1-h} ||L(U_L) - L(U_L^*)||_2\ , \tag{5}$$

where $U_L^* \in C^2(\Omega)$, $(x,y) \in \Omega$.

This theorem is proved in [7] for $\kappa = 0$. It can also be proved similarly for $\kappa > 0$.

Further we'll demonstrate the computations of all components of the right-hand side of (5) on a concrete example. But first let us consider the technique for finding an approximate solution of (1), (2).

3 Finding the Approximate Solution U_L^*

We shall find the approximate solution of (1), (2) as a series expansion over eigenfunctions of the Dirichlet problem for the Laplace operator

$$\Delta U = \lambda U, (x,y) \in \Omega\ , \tag{6}$$

$$U = 0, (x,y) \in \Gamma\ . \tag{7}$$

There exist countable sets of negative eigenvalues λ_i and eigenfunctions Φ_i of (6), (7) [8].

Thus, the following equalities hold true

$$\Delta \Phi_i = \lambda_i \Phi_i, (x,y) \in \Omega, i = 1, 2, \dots\ ,$$

$$\Phi_i = 0, (x,y) \in \Gamma, i = 1, 2, \dots\ .$$

It is also known [8] that the system $\Phi_i, i = 1, 2, \dots$ is an orthonormal basis in $\mathrm{L}^2(\Omega)$. That means, any function $U \in \mathrm{L}^2(\Omega)$ can be represented as a Fourier series converging in $\mathrm{L}^2(\Omega)$

$$U = \sum_{i=1}^{\infty} U_i \Phi_i,\ U_i = \langle U, \Phi_i \rangle_{\mathrm{L}^2(\Omega)}\ .$$

For operator L from (1), function Φ_i, and exact solution U_L of (1), (2) the Green's formula can be written

$$\iint\limits_{\Omega} U_L L(\Phi_i)\,dx\,dy = \iint\limits_{\Omega} \Phi_i L(U_L)\,dx\,dy,\ i = 1,2,\dots\ . \tag{8}$$

Since $\Phi_i(x,y)$ are the eigenfunctions of (6), (7) we obtain from (8) the following equalities

$$\iint\limits_{\Omega} U_L(\lambda_i\Phi_i - a(x,y)\Phi_i)\,dx\,dy = \iint\limits_{\Omega} \Phi_i f\,dx\,dy,\ i = 1,2,\dots\ . \tag{9}$$

We are looking for an approximate solution of (1), (2) as an expansion in a series over eigenfunctions of (6), (7)

$$U_L^* = \sum_{j=1}^{n} C_j\Phi_j\ . \tag{10}$$

Substituting (10) into the left side of (9) we obtain for first n equalities

$$\sum_{j=1}^{n} C_j \left(\iint\limits_{\Omega} \Phi_j(\lambda_i\Phi_i - a(x,y)\Phi_i)\,dx\,dy \right) = \iint\limits_{\Omega} \Phi_i f\,dx\,dy,\ i = 1,2,\dots,n\ . \tag{11}$$

Thus, we have a system of linear algebraic equations with respect to unknowns $C_j, j = 1,2,\dots,n$.

4 Calculation of $||G_{\Delta_\kappa;\Omega}(x,y;.,.)||_2$

Now let us consider the following concrete example of (1), (2)

$$L(U) \overset{def}{=} \Delta U - \beta\cos(x^2+y^2)U = f(x,y),\ (x,y)\in\Omega_1,\ \beta > 0\ , \tag{12}$$

$$U = 0,\ (x,y)\in\Gamma_1\ , \tag{13}$$

where

$$\Omega_1 = \{(x,y)\in\mathbb{R}^2 : x^2+y^2 < 1\},\ \Gamma_1 = \{(x,y)\in\mathbb{R}^2 : x^2+y^2 = 1\}\ ,$$

and

$$f(x,y) = 4\cos(1-x^2-y^2) + (4(x^2+y^2) + \beta\cos(x^2+y^2))\sin(1-x^2-y^2)\ .$$

The exact solution of (12), (13) is known

$$U_L(x,y) = \sin(x^2+y^2-1)\ .$$

and will be used to control further calculations.

As an auxiliary task we consider correspondingly the following Dirichlet problem for the Helmholtz equation

$$\Delta_\kappa(U) = \Delta U - \kappa U = f(x,y), (x,y) \in \Omega_1, \kappa > 0 \ , \tag{14}$$

$$U = 0, (x,y) \in \Gamma_1 \ . \tag{15}$$

The Green's function of (14), (15) can be given in polar coordinates explicitly as the following series [9]

$$G_{\Delta_\kappa;\Omega_1}(r,\varphi;\rho,\theta) = \sum_{l=1,2}\sum_{k=0}^{\infty}\sum_{j=1}^{\infty} \frac{w_{kj}^{(l)}(r,\varphi)w_{kj}^{(l)}(\rho,\theta)}{\lambda_{kj}^{*} - \kappa} \ ,$$

where $r,\rho \in [0,1),\ \varphi,\theta \in [0,2\pi)$, and

$$w_{0j}^{(1)}(r,\varphi) = \frac{1}{\sqrt{\pi}}\frac{1}{\left|J_0'\left(\mu_j^{(0)}\right)\right|}J_0'\left(\mu_j^{(0)}r\right), j = 1,2,\dots \ ,$$

$$w_{kj}^{(1)}(r,\varphi) = \frac{1}{\sqrt{\pi}}\frac{\sqrt{2}}{\left|J_k'\left(\mu_j^{(k)}\right)\right|}J_k'\left(\mu_j^{(k)}r\right)\cos(k\varphi),\ k,j = 1,2,\dots \ ,$$

$$w_{kj}^{(2)}(r,\varphi) = \frac{1}{\sqrt{\pi}}\frac{\sqrt{2}}{\left|J_k'\left(\mu_j^{(k)}\right)\right|}J_k'\left(\mu_j^{(k)}r\right)\sin(k\varphi),\ k,j = 1,2,\dots \ ,$$

$$\lambda_{kj}^{*} = -{\mu_j^{(k)}}^2 - \kappa,\ k = 0,1,\dots,\ j = 1,2,\dots \ .$$

are orthonormal eigenfunctions of (14), (15) and corresponding eigenvalues. Here $\mu_j^{(k)}$ denotes the j-th positive root of Bessel function $J_k(x)$.

So, in detail the Green's function of (14), (15) can be written as follows

$$G_{\Delta_\kappa;\Omega_1}(r,\varphi;\rho,\theta) = -\sum_{j=1}^{\infty}\frac{1}{\pi {J_0'}^2\left(\mu_j^{(0)}\right)}\frac{J_0\left(\mu_j^{(0)}r\right)J_0\left(\mu_j^{(0)}\rho\right)}{{\mu_j^{(0)}}^2 + 2\kappa} -$$

$$\sum_{k=1}^{\infty}\sum_{j=1}^{\infty}\frac{2(\cos(k\varphi)\cos(k\theta) + \sin(k\varphi)\sin(k\theta))}{\pi {J_k'}^2\left(\mu_j^{(k)}\right)}\frac{J_k\left(\mu_j^{(k)}r\right)J_k\left(\mu_j^{(k)}\rho\right)}{{\mu_j^{(k)}}^2 + 2\kappa} \ .$$

After raising this expression to the second power, integrating with respect to the variables (ρ,θ), and extracting the square root we obtain

$$||G_{\Delta_\kappa;\Omega_1}(r,\varphi;.,.)||_2 = \tag{16}$$

$$\frac{1}{\sqrt{\pi}}\sqrt{\sum_{j=1}^{\infty}\frac{1}{{J_0'}^2\left(\mu_j^{(0)}\right)}\frac{J_0^2\left(\mu_j^{(0)}r\right)}{\left({\mu_j^{(0)}}^2 + 2\kappa\right)^2} + 2\sum_{k=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{{J_k'}^2\left(\mu_j^{(k)}\right)}\frac{J_k^2\left(\mu_j^{(k)}r\right)}{\left({\mu_j^{(k)}}^2 + 2\kappa\right)^2}} \ .$$

To receive an approximate value of this expression one should evaluate partial sums of the series under the square root sign. We shall do this with *Mathematica* in several steps. First, we find the partial sum of the first series with `N0` items (we denote this partial sum by $\phi 1$).

```
In[1]:= << NumericalMath`BesselZeros`
       N0 = 50;   b0Z = BesselJZeros[0, N0];
       ϕ1 = Compile[{r, κ},  Evaluate[Sum[BesselJ[0, r*b0Z[[j]]]^2/
              ((D[BesselJ[0, x], x] /. x -> b0Z[[j]])^2*
               (b0Z[[j]]^2 + 2*κ)^2), {j, 1, N0}]]];
```

Then we form the procedure to calculate the partial sum $\phi 2$ of the second (double) series.

```
In[4]:= bZeros = Timing[Table[BesselJZeros[k, N0], {k, 1, N0}]];
       ϕ2 = Compile[{r, κ}, Evaluate[2*Sum[BesselJ[k, bZeros[[2, k, j]]*r]^2/
          ((D[BesselJ[k, x], x]  /. x -> bZeros[[2, k, j]])^2*
           (bZeros[[2, k, j]]^2+2*κ)^2), {k, 1, N0}, {j, 1, N0}]]];
```

So, the expression for the approximate value of the norm of the Green's function with respect to the second pair of variables in polar coordinates has the following form:

$$||G_{\Delta_\kappa;\Omega_1}(r,\varphi;.,.)||_2 \approx \frac{1}{\sqrt{\pi}}\sqrt{\phi 1+\phi 2}\ .$$

The norm of the Green's function doesn't involve the variable φ, as is seen in (16). So, we define the corresponding *Mathematica* subroutine as a function depending on r and κ.

```
In[6]:= normGreen[r_, κ_] := (1/Sqrt[Pi])*Sqrt[ϕ1[r, κ] + ϕ2[r, κ]];
```

5 Estimate of the Value of *h*

The value h^2 in the example under consideration is the following double integral which can be estimated and evaluated using (16):

$$h^2 = \iint\limits_{\Omega_1}\left(\iint\limits_{\Omega_1}\left|(\beta\cos(\xi^2+\eta^2)-\kappa)G_{\Delta_\kappa;\Omega_1}(x,y;\xi,\eta)\right|^2 d\xi\, d\eta\right)dx\, dy \le$$

$$(\beta^2-2\kappa\beta\cos(1)+\kappa^2)\iint\limits_{\Omega_1}\left(\iint\limits_{\Omega_1}G^2_{\Delta_\kappa;\Omega_1}(x,y;\xi,\eta)\, d\xi\, d\eta\right)dx\, dy =$$

$$2(\beta^2-2\kappa\beta\cos(1)+\kappa^2)\left(\sum_{j=1}^{\infty}\frac{1}{\left({\mu_j^{(0)}}^2+2\kappa\right)^2}+2\sum_{k=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{\left({\mu_j^{(k)}}^2+2\kappa\right)^2}\right)\ .$$

Taking into consideration the lower estimate for positive roots $\mu_m^{(k)}$ of Bessel function $J_k(x)$ from [10]

$$k+3(m-1)\le \mu_m^{(k)},\ m=1,2,\dots,\ k=0,1,2,\dots$$

we can write down another estimate for h^2:

$$h^2 \le 2(\beta^2-2\kappa\beta\cos(1)+\kappa^2)\times$$

$$\left(\sum_{j=1}^{\infty}\frac{1}{\left((3(j-1))^2+2\kappa\right)^2}+2\sum_{k=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{\left((k+3(j-1))^2+2\kappa\right)^2}\right)\ .$$

The first series from this expression can be calculated with *Mathematica.*

```
In[7]:= r1[κ_] = Simplify[Sum[1/((3*(i - 1))^2 + 2*κ)^2, {i, 1, ∞}],{κ > 0]}]
```

Out[7]= $\dfrac{2\pi^2\kappa\,\mathrm{csch}^2\left(\frac{1}{3}\sqrt{2}\pi\sqrt{\kappa}\right)+3\sqrt{2}\pi\sqrt{\kappa}\coth\left(\frac{1}{3}\sqrt{2}\pi\sqrt{\kappa}\right)+18}{144\kappa^2}$

The second (double) series can not be calculated directly. We find at first the sum with respect to j.

```
In[8]:= r2a[κ_] = TrigToExp[FullSimplify[Sum[1/(k + (3*(j - 1))^2 + 2*κ)^2,
           {j, 1, ∞}], {k > 0, k∈Integers, κ > 0, κ ∈ Reals}]]
```

Out[8]= $\left(\frac{3}{2}\pi\sqrt{k+2\kappa}\left(-e^{-\frac{2}{3}\pi\sqrt{k+2\kappa}}+e^{\frac{2}{3}\pi\sqrt{k+2\kappa}}\right)+9\left(e^{-\frac{2}{3}\pi\sqrt{k+2\kappa}}+e^{\frac{2}{3}\pi\sqrt{k+2\kappa}}\right)+2\pi^2(k+2\kappa)-18\right)\Big/\left(18\left(-e^{-\frac{1}{3}\pi\sqrt{k+2\kappa}}+e^{\frac{1}{3}\pi\sqrt{k+2\kappa}}\right)^2(k+2\kappa)^2\right)$ It is simply

to prove that the following estimates hold true:

$$\frac{1}{\left(-e^{-\frac{1}{3}\pi\sqrt{k+2\kappa}}+e^{\frac{1}{3}\pi\sqrt{k+2\kappa}}\right)^2}\le\frac{e^{2\pi/3}}{\left(e^{2\pi/3}-1\right)^2}\overset{def}{=}E_1\ ,$$

$$\frac{\left(e^{-\frac{2}{3}\pi\sqrt{k+2\kappa}}+e^{\frac{2}{3}\pi\sqrt{k+2\kappa}}\right)}{\left(-e^{-\frac{1}{3}\pi\sqrt{k+2\kappa}}+e^{\frac{1}{3}\pi\sqrt{k+2\kappa}}\right)^2}\le\frac{1+e^{4\pi/3}}{\left(e^{2\pi/3}-1\right)^2}\overset{def}{=}E_2\ ,$$

$$\frac{\left(-e^{-\frac{2}{3}\pi\sqrt{k+2\kappa}}+e^{\frac{2}{3}\pi\sqrt{k+2\kappa}}\right)}{\left(-e^{-\frac{1}{3}\pi\sqrt{k+2\kappa}}+e^{\frac{1}{3}\pi\sqrt{k+2\kappa}}\right)^2}\le\frac{1+e^{2\pi/3}}{e^{2\pi/3}-1}\overset{def}{=}E_3\ ,$$

$$\frac{k+2\kappa}{\left(-e^{-\frac{1}{3}\pi\sqrt{k+2\kappa}}+e^{\frac{1}{3}\pi\sqrt{k+2\kappa}}\right)^2}\le\frac{e^{2\pi/3}}{\left(e^{2\pi/3}-1\right)^2}\overset{def}{=}E_4\ .$$

Hence, we can finally calculate the upper estimate for the double sum:

$$\sum_{k=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{\left((k+3(j-1))^2+2\kappa\right)^2}\leq$$

$$\sum_{k=1}^{\infty}\frac{E_1}{(k+2\kappa)^2}+\sum_{k=1}^{\infty}\frac{E_2}{2(k+2\kappa)^2}+\sum_{k=1}^{\infty}\frac{E_3\pi\sqrt{k+2\kappa}}{12(k+2\kappa)^2}+\sum_{k=1}^{\infty}\frac{E_4\pi^2}{9(k+2\kappa)^2}\quad .$$

```
In[9]:= E1 = E^((2*Pi)/3)/(-1 + E^((2*Pi)/3))^2;
        E2 = (1 + E^((4*Pi)/3))/(-1 + E^((2*Pi)/3))^2;
        E3 = (1 +E^((2*Pi)/3))/(-1 + E^((2*Pi)/3));
        E4 = E^((2*Pi)/3)/(-1 + E^((2*Pi)/3))^2;
```

```
In[13]:= r2[κ_] = Simplify[ Sum[(-18*E1)/(18*(k + 2*κ)^2),{k, 1, ∞}] +
             Sum[(9*E2)/(18*(k + 2*κ)^2), {k, 1, ∞}] +
             Sum[((3/2)*E3*Pi*Sqrt[k + 2*κ])/(18*(k + 2*κ)^2), {k, 1, ∞}] +
             Sum[(2*Pi^2*E4)/(18*(k + 2*κ)^2), {k, 1, ∞}]]
```

Out[13]= $$\frac{2\left(2e^{2\pi/3}(\pi^2-9)+9e^{4\pi/3}+9\right)\psi^{(1)}(2\kappa+1)+3\left(e^{4\pi/3}-1\right)\pi\zeta\left(\frac{3}{2},2\kappa+1\right)}{36(e^{2\pi/3}-1)^2}$$

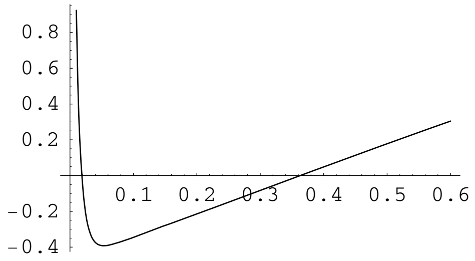
Now we evaluate the whole upper estimate for h.

```
In[14]:= hEstimate[β_, κ_] = FullSimplify[ Sqrt[2*(β^2 -
             2*κ*β*Cos[1] + κ^2)*(r1[κ] + 2*r2[κ])], {κ > 0, β > 0}]
```

Out[14]= $$\frac{\sqrt{\beta^2-2\kappa\cos(1)\beta+\kappa^2}}{6\sqrt{2}\kappa}\times$$
$$\left(8\left(\left(\pi^2\operatorname{csch}^2(\frac{\pi}{3})+18\right)\psi^{(1)}(2\kappa+1)+3\pi\coth(\frac{\pi}{3})\zeta(\frac{3}{2},2\kappa+1)\right)\kappa^2+\operatorname{csch}^2\left(\frac{1}{3}\sqrt{2}\pi\sqrt{\kappa}\right)\kappa+3\sqrt{2}\coth\left(\frac{1}{3}\sqrt{2}\pi\sqrt{\kappa}\right)\sqrt{\kappa}+18\right)^{1/2}$$

To be able to use the error estimate (5) we should find those values of κ which deliver negative values of the expression $h-1$. For example, for $\beta=1/32$ this expression has the following graphical representation.

```
In[15]:= Plot[hEstimate[1/32, κ] - 1, {κ, 0.01, 0.6}];
```



In[16]:=
```
FindRoot[hEstimate[1/32, κ] - 1 == 0, {κ, 0.01}][[1, 2]]
FindRoot[hEstimate[1/32, κ] - 1 == 0, {κ, 0.5}][[1, 2]]
```

Out[16]= 0.0186874
0.363051

So, for all value of κ between 0.0186874 and 0.363051 the inequality $h-1<0$ holds true. We can also find a point, where the value of $h-1$ is minimal. That means, for such a value of κ we shall obtain the best error estimate according (5).

In[18]:=
```
κOpt = FindRoot[D[hEstimate[1/32, κ], κ] == 0, {κ, 0.1}][[1, 2]]
```

Out[18]= 0.0534114

6 Computational Procedure for Finding the Approximate Solution of (12), (13)

We fix, at first, the finite number of the eigenfunctions $w_{0j}^{(1)}$, $w_{kj}^{(1)}$, $w_{kj}^{(2)}$ of (14), (15).

In[19]:=
```
n=4; n1=2; n2=2; m1=2; m2=2;
```

We take, for example, `n` functions from the first group $w_{0j}^{(1)}$, `n1`×`n2` functions from the second group $w_{kj}^{(1)}$, and `m1`×`m2` functions from the third group $w_{kj}^{(2)}$. Then three lists of orthonormal eigenfunctions are formed.

In[20]:=
```
tb0 = Table[BesselJ[0, r*BesselJZeros[0, n][[j]]]/
    (Sqrt[Pi]*Abs[D[BesselJ[0, x], x] /.
       x -> BesselJZeros[0, n][[j]]]), {j, 1, n}]
```

Out[20]= $\{1.08676\,J_0(2.40483\,r), 1.65809\,J_0(5.52008\,r), 2.07841\,J_0(8.65373\,r), 2.42704\,J_0(11.7915\,r)\}$

In[21]:=
```
tb1 = Table[(Sqrt[2]*BesselJ[k, r*BesselJZeros[k, n2][[j]]]*
     Cos[k*φ])/(Sqrt[Pi]*Abs[D[BesselJ[k, x], x] /.
       x -> BesselJZeros[k, n2][[j]]]), {k, 1, n1}, {j, 1, n2}]
```

Out[21]= $\begin{pmatrix} 1.98105\,J_1(3.83171\,r)\cos(\varphi) & 2.65859\,J_1(7.01559\,r)\cos(\varphi) \\ 2.34901\,J_2(5.13562\,r)\cos(2\varphi) & 2.94007\,J_2(8.41724\,r)\cos(2\varphi) \end{pmatrix}$

In[22]:=
```
tb2 = Table[(Sqrt[2]*BesselJ[k, r*BesselJZeros[k, m2][[j]]]*
     Sin[k*φ])/(Sqrt[Pi]*Abs[D[BesselJ[k, x], x] /.
       x -> BesselJZeros[k, m2][[j]]]), {k, 1, m1}, {j, 1, m2}]
```

Out[22]= $\begin{pmatrix} 1.98105\,J_1(3.83171\,r)\sin(\varphi) & 2.65859\,J_1(7.01559\,r)\sin(\varphi) \\ 2.34901\,J_2(5.13562\,r)\sin(2\varphi) & 2.94007\,J_2(8.41724\,r)\sin(2\varphi) \end{pmatrix}$

The lists of orthonormal eigenfunctions are combined into one-dimensional list.

In[23]:= `tb = Join[tb0, tb1, tb2] // Flatten;`

Now we check that all functions thus calculated are orthonormal in L^2.

In[24]:=
```
MatrixForm[ Chop[ Table[ Integrate[
   r*Integrate[tb[[i]]*tb[[j]], {φ, 0, 2*Pi}], {r, 0, 1}],
     {i, Length[tb]}, {j, Length[tb]}]]]
```

Out[24]=
$$\begin{pmatrix}1. & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1. & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1. & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1. & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1. & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1. & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1. & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.\end{pmatrix}$$

Also we form three lists of the roots of Bessel functions for each group of eigenfunctions and then combine them into one list.

In[25]:=
```
μ0 = Table[BesselJZeros[0, n][[j]], {j, 1, n}];
μ1 = Table[BesselJZeros[k, n2][[j]], {k, 1, n1}, {j, 1, n2}];
μ2 = Table[BesselJZeros[k, m2][[j]], {k, 1, m1}, {j, 1, m2}];
μ = Join[mu0, mu1, mu2] // Flatten
```

Out[28]= $\{2.40483, 5.52008, 8.65373, 11.7915, 3.83171, 7.01559, 5.13562, 8.41724, 3.83171, 7.01559, 5.13562, 8.41724\}$

Now we define the expression for the approximate solution of (12), (13).

In[29]:= `cc = Array[c, Length[tb]];  Uapp = cc.tb // Short`

Out[29]= $1.08676\,J_0(2.40483\,r)c(1) + \ll 10 \gg + 2.94007\,J_2(8.41724\,r)c(12)\sin(2\varphi)$

Here $c(1), \ldots, c(12)$ are unknown constants mentioned in (10).

Further, we define in *Mathematica* the concrete functions $a(x, y)$ and $f(x, y)$.

In[30]:=
```
f[x_, y_] = (1/32)*(128*Cos[1 - x^2 - y^2] +
    (128*(x^2 + y^2) + Cos[x^2 + y^2])*Sin[1 - x^2 - y^2]);
a[x_, y_] = Cos[x^2 + y^2]/32;
```

We describe also the lists of integrands for the left- and right-hand side of linear algebraic system defined in general form by (11).

In[32]:=
```
rigths = Table[tb[[i]]*f[x, y], {i, Length[tb]}] /.
  {x -> r*Cos[φ], y -> r*Sin[φ]};
lefts = Table[Uapp*((-mu[[i]]^2)*tb[[i]] -
  Simplify[a[r*Cos[φ], r*Sin[φ]]]*tb[[i]]), {i, Length[tb]}];
```

The integrands for the left-hand side of the system still include the unknowns $c(1), c(2), \ldots, c(12)$ looked for. So, we should separate these unknowns from their coefficients.

```
In[34]:= TempMatr = Table[Coefficient[Expand[lefts[[i]]], c[j]],
             {i, Length[lefts]}, {j, Length[cc]}];
```

We calculate now the integrals of all elements of the matrix thus obtained and receive the matrix of a linear algebraic system with respect to unknown expansion coefficients.

```
In[35]:= Matr = Table[NIntegrate[r*Integrate[TempMatr[[i, j]],
           {φ, 0, 2*Pi}], {r, 0, 1}, AccuracyGoal -> 15],
                {i, Length[lefts]}, {j, Length[cc]}];
         Short[Matr, 6]
```

```
Out[36]= {{−5.81322, −0.00151404, 0.00082594, −0.00032907, 0., 0., 0., 0., 0., 0., 0., 0.},
          ≪10≫, {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., −0.0021503, −70.8782}}
```

In a similar way, we calculate the integrals of all elements of expression vector for the right-hand side of the system.

```
In[37]:= RSide = Table[NIntegrate[r*Integrate[rigths[[i]],
           {φ, 0, 2*Pi}], {r, 0, 1}, AccuracyGoal -> 15], {i, Length[rigths]}]
```

```
Out[37]= {5.33393, −3.58757, 1.97892, −1.34107, 0., 0., 0., 0., 0., 0., 0., 0.}
```

We solve the system and write down the formula for the approximate solution of (12), (13).

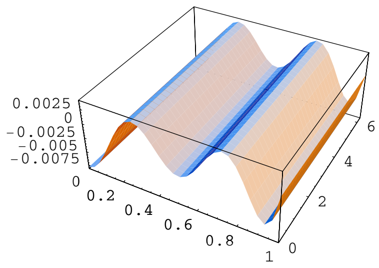
```
In[38]:= coeffs = LinearSolve[Matr, RSide]
```

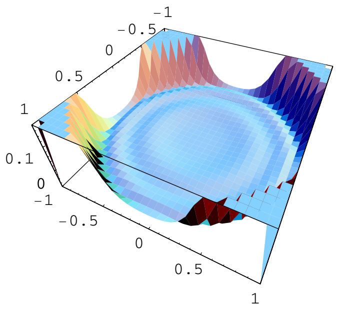
```
Out[38]= {−0.917587, 0.117673, −0.026429, 0.00964673, 0., 0., 0., 0., 0., 0., 0., 0.}
```

```
In[39]:= appSol[r_, φ_] = coeffs.tb // Chop
```

Out[39]=
$$-0.997198\,J_0(2.40483\,r) + 0.195112\,J_0(5.52008\,r) - 0.0549304\,J_0(8.65373\,r) + 0.023413\,J_0(11.7915\,r)$$

The difference between the exact solution and the approximate one thus obtained can be represented graphically both in polar and Cartesian coordinates:





7 Evaluation of $||f - L(U_L^*)||_2$ and the Error Estimate

Let us define the operator L and the right-hand side of (12) in polar coordinates.

In[40]:=
```
LPol[W_, {r_, φ_}] := (1/r)*D[r*D[W, r], r] +
          (1/r^2)*D[W, φ, φ] - (Cos[r^2]/32)*W
```

In[41]:=
```
fPol[r_, φ_] = Simplify[f[x,y] /. {x -> r*Sin[φ], y -> r*Cos[φ]}]
```

Out[41]= $\frac{1}{32}\left(128\cos\left(1-r^2\right)+\left(128r^2+\cos\left(r^2\right)\right)\sin\left(1-r^2\right)\right)$

The defect $f - L(U_L^*)$ will have then the following form.

In[42]:=
```
defect[r_, φ_] = LPol[appSol[r, φ], {r, φ}] - fPol[r, φ]
```

Out[42]=
$$\frac{1}{r}(2.39809 J_1(2.40483r) - 1.07703 J_1(5.52008r) + 0.475352 J_1(8.65373r) - 0.276075\, J_1(11.7915\, r) + r(2.88349(J_0(2.40483\, r) - J_2(2.40483\, r)) - 2.97266(J_0(5.52008\, r) - J_2(5.52008\, r)) + 2.05679(J_0(8.65373\, r) - J_2(8.65373\, r)) - 1.62768(J_0(11.7915\, r) - J_2(11.7915\, r)))) - \frac{1}{32}(-0.997198\, J_0(2.40483\, r) + 0.195112\, J_0(5.52008\, r) - 0.0549304\, J_0(8.65373\, r) + 0.023413\, J_0(11.7915\, r))\cos\left(r^2\right) + \frac{1}{32}\left(-128\cos\left(1-r^2\right) - \left(128r^2+\cos\left(r^2\right)\right)\sin\left(1-r^2\right)\right)$$

Now we calculate the defect norm:

In[43]:=
```
nDef = Sqrt[NIntegrate[r*Integrate[defect[r, φ]^2,
    {φ, 0, 2*Pi}], {r, 0, 1}]]
```

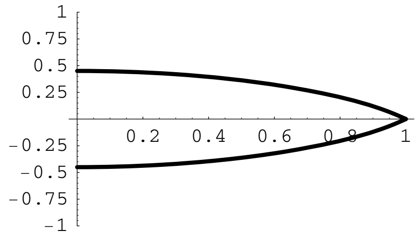
Out[43]= 2.24889

So, now we have all constituents to evaluate the upper estimate of the right-hand side of (5).

In[44]:=
```
rhs[r_, κ_] :=normGreen[r, κ]/(1 - hEstimate[1/32, κ])*nDef
```

This *Mathematica* function as well as the approximate solution found in section 6 doesn't depend of the variable φ. Therefore, we can graphically illustrate the formula (5) in polar coordinates using two-dimensional plot:

In[45]:=
```
Plot[{rhs[r,κOpt], -rhs[r, κOpt]}, {r, 0, 1}];
```



Strictly speaking, before to use the uniform estimate (5) in practice it is necessary to consider more closely the representation of the norm of the Green's function (16). We have calculated this expression approximately, as a combination of partial sums, and, therefore, one should still estimate the series remainder in this formula. Then, according to (5) the error of the approximate solution of (12), (13) would be guaranteed to lie within bounds indicated on the latter plot.

References

1. Wolfram, S.: The Mathematica Book, 4th Edition. Champain, IL, Wolfram Media (1999)
2. Dzyadyk, V.K.: Approximated methods for solving differential and integral equations. Nauk. dumka, Kiev (1988)
3. Hantzschmann, K.: Zur Lösung von Randwertaufgaben bei Systemen gewönlichen Deifferentialgleichungen mit dem Ritz-Galerkin-Verfahren. Habilitationsschrift Technische Universitet Dresden (1983)
4. Lehmann, N.J.: Fehlerschranken für Näherungslösungen bei Differentialgleichungen. Numerische Mathematik **10** (1967) 261–288
5. Becken, O., Jung, A.: Error estimation in the case of linear ordinary differential equations. Rostoker Informatik-Berichte **22** (1998)
6. Rösler, T.: Adaptiv-iteratives Verfahren zur Lsung von Differenzialgleichungen. Rostoker Informatik-Berichte **28** (2003) 89–108
7. Zemskov S.: The Error Estimate of the Approximate Solution of the Dirichlet Problem for Elliptical Partial Differential Equation. Computer Algebra in Scientific Computing, Proceedings of the Seventh International Workshop, Technische Universität München (2004) 479–484
8. Mihajlov V.P.: Partial differential equations. Nauka, Moskva (1976) (in Russian)
9. Polianin A.D.: Handbook of Linear Partial Differential Equations for Engineers and Scientists. Chapman & Hill/CRC (2002)
10. Koshliakov, N.S., Gliner, E.B., Smirnov, M.M.: Differential equations of mathematical physics. Moskva, State Publishing House for physical and mathematical literature (1962) (in Russian)

Author Index