

Vladimir P. Gerdt
Ernst W. Mayr
Evgenii V. Vorozhtsov (Eds.)

LNCS 5743

Computer Algebra in Scientific Computing

11th International Workshop, CASC 2009
Kobe, Japan, September 2009
Proceedings

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Vladimir P. Gerdt Ernst W. Mayr
Evgenii V. Vorozhtsov (Eds.)

Computer Algebra in Scientific Computing

11th International Workshop, CASC 2009
Kobe, Japan, September 13-17, 2009
Proceedings

Volume Editors

Vladimir P. Gerdt

Joint Institute for Nuclear Research

Laboratory of Information Technologies, Dubna, Russia

E-mail: gerdt@jinr.ru

Ernst W. Mayr

Technische Universität München

Institut für Informatik

Garching, Germany

E-mail: 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: 2009933206

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

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743

ISBN-10 3-642-04102-7 Springer Berlin Heidelberg New York

ISBN-13 978-3-642-04102-0 Springer Berlin Heidelberg New York

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

springer.com

© Springer-Verlag Berlin Heidelberg 2009

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper SPIN: 12747016 06/3180 5 4 3 2 1 0

Preface

After a pause taken in 2008 (owed to the political situation in the Caucasus region where CASC 2008 was supposed to take place), CASC 2009 continued the series of international workshops on the latest advances and trends both in the methods of computer algebra and in applications of computer algebra systems (CASs) to the solution of various problems in scientific computing.

The ten earlier CASC conferences, CASC 1998, CASC 1999, CASC 2000, CASC 2001, CASC 2002, CASC 2003, CASC 2004, CASC 2005, CASC 2006, and CASC 2007 were held, respectively, in St. Petersburg (Russia), in Munich (Germany), in Samarkand (Uzbekistan), in Konstanz (Germany), in Yalta (Ukraine), in Passau (Germany), in St. Petersburg (Russia), in Kalamata (Greece), in Chişinău (Moldova), and in Bonn, Germany, and they all proved to be successful.

Research in the area of computer algebra and its applications is being conducted actively in Japan. For example, during the years 2005–2007, at the CASC conferences, talks were presented by researchers from the University of Kobe, Osaka University, Yamaguchi University, Kyushu University, University of Tsukuba, Waseda University, Cybernet Systems, Japan Science and Technology Agency, and several other institutions of Japan. In this connection, it was decided to hold the CASC 2009 Workshop in Japan in the hope that it would help bring together non-Japanese and Japanese researchers working both in the areas of computer algebra (CA) methods and of various CA applications in natural sciences and engineering.

The present volume contains revised versions of the papers submitted to the workshop by the participants and accepted by the Program Committee after a thorough reviewing process (each paper was reviewed by at least three referees).

A number of papers included in the proceedings deal with computer algebra methods and algorithms: these are contributions to the computation of Groebner bases, to quantifier elimination, to number theory, to the theory of matrices, to polynomial algebra, and to the analytical solution of linear difference equations.

Several papers are devoted to the application of symbolic manipulations for obtaining new analytic solutions of both ordinary and linear and nonlinear partial differential equations.

There are also papers in which the CA methods and results are applied for the derivation of new interesting finite-difference schemes for numerical integration of partial differential equations of mathematical physics, including the two-dimensional Navier–Stokes equations.

The presented applications of computer algebra techniques and CASs include tasks in the fields of fluid mechanics, nuclear physics, quantum mechanics, stability of satellites, dimensionality theory, discrete dynamics, and epidemic modeling.

In the invited lecture of X.-S. Gao et al., an important task of triangular meshing of implicit algebraic surfaces with singularities was considered. Such

tasks are basic operations in computer graphics, geometric modeling, and finite-element methods. To solve the above task the authors propose to use an isotopic meshing, whose feature is that it possesses correct topology. Symbolic computations are used to guarantee the correctness, and numerical computations are used whenever possible to enhance the efficiency.

The other invited talk authored by J.-Ch. Faugère dealt with algebraic cryptanalysis as a general method for solving the problem of evaluating the security of cryptosystems, which is of fundamental importance in cryptography. Within this approach, one has to solve a set of multivariate polynomial equations. It is shown that for a recommended family of parameters, one can solve the corresponding systems in polynomial time and, thus, break the corresponding cryptosystem.

The CASC 2009 workshop was supported financially by the Japan Society for the Promotion of Science (JSPS), Cybernet Systems, and Maplesoft. The JSPS also supported financially the publication of the present proceedings volume. Our particular thanks are due to the members of the CASC 2009 Local Organizing Committee in Japan: K. Nagasaka (Kobe University), T. Kitamoto (Yamaguchi University), and T. Yamaguchi (Cybernet Systems), who ably handled local arrangements in Kobe. We are grateful to W. Meixner for his technical help in the preparation of the camera-ready manuscript for this volume.

July 2009

V.P. Gerdt
E.W. Mayr
E.V. Vorozhtsov

Organization

CASC 2009 was organized jointly by the Department of Informatics at the Technische Universität München, Germany, and the Graduate School of Human Development and Environment, Kobe University, Japan.

Workshop General Chairs

Vladimir Gerdt (JINR, Dubna)

Ernst W. Mayr (TU München)

Program Committee Chair

Evgenii Vorozhtsov (Novosibirsk)

Program Committee

Alkis Akritas (Volos)

Gerd Baumann (Cairo)

Hans-Joachim Bungartz (Munich)

Andreas Dolzmann (Passau)

Victor Edneral (Moscow)

Ioannis Emiris (Athens)

Jaime Gutierrez (Santander)

Robert Kragler (Weingarten)

Richard Liska (Prague)

Marc Moreno Maza (London, Ontario)

Eugenio Roanes-Lozano (Madrid)

Markus Rosenkranz (Linz)

Mohab Safey El Din (Paris)

Yosuke Sato (Tokyo)

Werner Seiler (Kassel)

Doru Stefanescu (Bucharest)

Stanly Steinberg (Albuquerque)

Serguei P. Tsarev (Krasnoyarsk)

Andreas Weber (Bonn)

Song Yan (Cambridge, Mass.)

Local Organization

K. Nagasaka (Kobe University)

T. Ymaguchi (Cybernet Systems)

T. Kitamoto (Ymaguchi University)

Publicity Chair

Victor G. Ganzha (Munich)

Website

<http://wwwmayr.in.tum.de/CASC2009>

Table of Contents

On m -Interlacing Solutions of Linear Difference Equations	1
<i>S.A. Abramov, M.A. Barkatou, and D.E. Khmelnov</i>	
Parametric Analysis of Stability Conditions for a Satellite with Gyrodines	18
<i>Andrey V. Banshchikov</i>	
Computing and Visualizing Closure Objects Using Relation Algebra and RELVIEW	29
<i>Rudolf Berghammer and Bernd Braßel</i>	
On Integrability of a Planar ODE System Near a Degenerate Stationary Point	45
<i>Alexander Bruno and Victor Edneral</i>	
Conditions of D-Stability of the Fifth-Order Matrices	54
<i>Larisa A. Burlakova</i>	
Code Generation for Polynomial Multiplication	66
<i>Ling Ding and Éric Schost</i>	
Solving Structured Polynomial Systems and Applications to Cryptology	79
<i>Jean-Charles Faugère</i>	
The Comparison Method of Physical Quantity Dimensionalities	81
<i>Alexander V. Flegontov and M.J. Marusina</i>	
Ambient Isotopic Meshing for Implicit Algebraic Surfaces with Singularities (An Extended Abstract)	89
<i>Jin-San Cheng, Xiao-Shan Gao, and Jia Li</i>	
Involution and Difference Schemes for the Navier–Stokes Equations	94
<i>Vladimir P. Gerdt and Yuri A. Blinkov</i>	
A Mathematica Package for Simulation of Quantum Computation	106
<i>Vladimir P. Gerdt, Robert Kragler, and Alexander N. Prokopenya</i>	
On Computing the Hermite Form of a Matrix of Differential Polynomials	118
<i>Mark Giesbrecht and Myung Sub Kim</i>	
On the Computation of Comprehensive Boolean Gröbner Bases	130
<i>Shutaro Inoue</i>	

On Invariant Manifolds of Dynamical Systems in Lie Algebras	142
<i>Valentin Irtegov and Tatyana Titorenko</i>	
On the Complexity of Reliable Root Approximation	155
<i>Michael Kerber</i>	
Algebraic Approach to the Computation of the Defining Polynomial of the Algebraic Riccati Equation	168
<i>Takuya Kitamoto</i>	
Discrete Dynamics: Gauge Invariance and Quantization	180
<i>Vladimir V. Kornyak</i>	
Effective Quantifier Elimination for Presburger Arithmetic with Infinity	195
<i>Aless Lasaruk and Thomas Sturm</i>	
An Algorithm for Symbolic Solving of Differential Equations and Estimation of Accuracy	213
<i>Natasha Malaschonok</i>	
Lazy and Forgetful Polynomial Arithmetic and Applications	226
<i>Michael Monagan and Paul Vrbik</i>	
On the Average Growth Rate of Random Compositions of Fibonacci and Padovan Recurrences	240
<i>Nikita Gogin and Aleksandr Mylläri</i>	
A Study on Gröbner Basis with Inexact Input	247
<i>Kosaku Nagasaka</i>	
Modular Algorithms for Computing a Generating Set of the Syzygy Module	259
<i>Masayuki Noro</i>	
A Symbolic Framework for Operations on Linear Boundary Problems . . .	269
<i>Markus Rosenkranz, Georg Regensburger, Loredana Tec, and Bruno Buchberger</i>	
Mathematical Model for Dengue Epidemics with Differential Susceptibility and Asymptomatic Patients Using Computer Algebra . . .	284
<i>Clarita Saldarriaga Vargas</i>	
Multiple Factorizations of Bivariate Linear Partial Differential Operators	299
<i>Ekaterina Shemyakova</i>	
Computing Gröbner Bases within Linear Algebra	310
<i>Akira Suzuki</i>	

A Mimetic Finite-Difference Scheme for Convection of Multicomponent Fluid in a Porous Medium	322
<i>Vyacheslav Tsybulin, Andrew Nemtsev, and Bülent Karasözen</i>	
Symbolic-Numerical Algorithms for Solving Parabolic Quantum Well Problem with Hydrogen-Like Impurity	334
<i>S.I. Vinitsky, O. Chuluunbaatar, V.P. Gerdt, A.A. Gusev, and V.A. Rostovtsev</i>	
New Analytic Solutions of the Problem of Gas Flow in a Casing with Rotating Disc	350
<i>Evgenii V. Vorozhtsov</i>	
Hybrid Solution of Two-Point Linear Boundary Value Problems	373
<i>M. Youssef and G. Baumann</i>	
Author Index	393

On m -Interlacing Solutions of Linear Difference Equations^{*}

S.A. Abramov¹, M.A. Barkatou², and D.E. Khmelnov³

¹ Computing Centre of the Russian Academy of Sciences, Vavilova, 40,
Moscow 119991, GSP-1 Russia

`sergeyabramov@mail.ru`

² Institute XLIM, Université de Limoges, CNRS, 123, Av. A. Thomas, 87060
Limoges cedex, France

`moulay.barkatou@unilim.fr`

³ Computing Centre of the Russian Academy of Sciences, Vavilova, 40,
Moscow 119991, GSP-1, Russia

`dennis.khmelnov@mail.ru`

Abstract. We consider linear homogeneous difference equations with rational-function coefficients. The search for solutions in the form of the m -interlacing ($1 \leq m \leq \text{ord } L$, where L is a given operator) of finite sums of hypergeometric sequences, plays an important role in the Hendriks–Singer algorithm for constructing all Liouvillian solutions of $L(y) = 0$. We show that Hendriks–Singer’s procedure for finding solutions in the form of such m -interlacing can be simplified. We also show that the space of solutions of $L(y) = 0$ spanned by the solutions of the form of the m -interlacing of hypergeometric sequences possesses a cyclic permutation property. In addition, we describe adjustments of our implementation of the Hendriks–Singer algorithm to utilize the presented results.

1 Introduction

In [5] the definition of Liouvillian sequence was given, and the fact that the Galois group of a linear homogeneous difference equation with rational-function coefficients is solvable iff the equation has a fundamental system of Liouvillian solutions was proven (the definition of the Galois group of an equation of this type was given earlier in [10]). Let \mathcal{C} be an algebraically closed subfield of the field \mathbb{C} of complex numbers. In [5] two sequences $u, v : \mathbb{N} \rightarrow \mathcal{C}$ are supposed to be equal iff $u_n = v_n$ for all integer n large enough, i.e., factually the germs of sequences are considered. The ring of the germs of sequences is denoted by \mathcal{S} . As it is done in [5], we will frequently identify a sequence with its equivalent class in \mathcal{S} . We will use the symbol $\bar{\forall}n$ as “for all integer n large enough”. Denote $k = \mathcal{C}(x)$. This field can be embedded in \mathcal{S} : since the germs of sequences are considered we can map $f \in k$, e.g., to the sequence u such that $u_n = 0$ if n is a pole of f and $f(n)$ otherwise.

^{*} Supported by ECONET grant 21315ZF.

The map $\phi : \mathcal{S} \rightarrow \mathcal{S}$ defined by $\phi(u_0, u_1, u_2, \dots) = (u_1, u_2, u_3, \dots)$ is an automorphism of \mathcal{S} (with $\phi(f(x)) = f(x + 1)$ for $f(x) \in k$). We say that a sequence $u = \langle u_n \rangle$ satisfies the equation $L(y) = 0$ with

$$L = \phi^d + a_{d-1}(x)\phi^{d-1} + \dots + a_1(x)\phi + a_0(x), \tag{1}$$

$a_1(x), a_2(x), \dots, a_{d-1}(x) \in k, a_0(x) \in k \setminus \{0\}$, if the sequence

$$\langle u_{n+d} + a_{d-1}(n)u_{n+d-1} + \dots + a_1(n)u_{n+1} + a_0(n)u_n \rangle$$

is equal to zero sequence, i.e., if

$$u_{n+d} + a_{d-1}(n)u_{n+d-1} + \dots + a_1(n)u_{n+1} + a_0(n)u_n = 0, \quad \forall n.$$

For short, we will talk about solutions of L instead of solutions of the equation $L(y) = 0$.

The definition of Liouvillian sequence (we will discuss this definition in Section 5.1) uses the notion of the *interlacing* of sequences: for sequences $b^{(0)} = \langle b_n^{(0)} \rangle, b^{(1)} = \langle b_n^{(1)} \rangle, \dots, b^{(m-1)} = \langle b_n^{(m-1)} \rangle, m \geq 1$, their interlacing is the sequence $u = \langle u_n \rangle$ defined by

$$u_n = \begin{cases} b_{\frac{n}{m}}^{(0)}, & \text{if } n \equiv 0 \pmod{m}, \\ b_{\frac{n-1}{m}}^{(1)}, & \text{if } n \equiv 1 \pmod{m}, \\ \dots\dots\dots \\ b_{\frac{n-m+1}{m}}^{(m-1)}, & \text{if } n \equiv m-1 \pmod{m}. \end{cases} \tag{2}$$

For example, the interlacing of two sequences

$$\begin{aligned} b^{(0)} &: b_0^{(0)}, b_1^{(0)}, b_2^{(0)}, \dots, \\ b^{(1)} &: b_0^{(1)}, b_1^{(1)}, b_2^{(1)}, \dots \end{aligned}$$

is the sequence u_0, u_1, u_2, \dots of the form

$$b_0^{(0)}, b_0^{(1)}, b_1^{(0)}, b_1^{(1)}, b_2^{(0)}, b_2^{(1)}, \dots$$

A non-zero sequence g is *hypergeometric* if it satisfies a first order operator

$$\phi - h(x), \quad h(x) \in k,$$

the rational function $h(x)$ is the *certificate* of g . By definition, zero sequence is also hypergeometric with zero certificate.

The interlacing of m sequences, $m \geq 1$, which have the form of finite sums of hypergeometric sequences will be called an m -interlacing.

The \mathcal{C} -linear spaces of all sequences that satisfy L and, resp., of all m -interlacings, that satisfy L will be denoted by $V(L)$ and, resp., $V_m(L), m \geq 1$. The Hendriks–Singer algorithm (HS) for constructing a basis for the \mathcal{C} -linear space of Liouvillian solutions of L is based on two facts proven in [5]:

(a) If L has a Liouvillian solution then for some integer m , $1 \leq m \leq \text{ord } L$, the operator L has a solution in the form of an m -interlacing.

(b) For any integer m , $1 \leq m \leq \text{ord } L$, one can construct algorithmically an operator $H \in k[\phi]$ such that $V(H) = V_m(H) = V_m(L)$. (It is possible, of course, that $\text{ord } H = \dim V_m(H) = 0$.)

The central part of HS is constructing for a given m the operator H mentioned in (b), and a basis for $V_m(H)$. This procedure (a part of HS) will be denoted by mHS.

In Section 2, a simplification of the procedure mHS by removing some unnecessary actions is described. (The authors of the paper [5] notice that they ignore effectiveness questions and just try to present their algorithm in an understandable form — see Remarks on p. 251 of [5].)

In Section 3, we briefly consider the special cases when \mathcal{C} is not algebraically closed, and the case of an irreducible L (the Cha – van Hoeij algorithm [2]).

In Section 4, we prove some properties of the space $V_m(L)$.

In Section 5, an implementation of a simplified version of mHS and a modified version of the search for all Liouvillian solutions is described.

2 Search for m -Interlacing Solutions of L for a Fixed m

2.1 The Hendriks–Singer Procedure for Finding m -Interlacing Solutions

Let L be of the form (1), an integer m such that $1 \leq m \leq d$ be fixed, and $\tau : k \rightarrow k$ be the automorphism defined by $x \mapsto mx$. The procedure described in [5] by Hendriks and Singer for finding all solutions of L which have the form of m -interlacings is as follows.

mHS₁: Constructing a polynomial $P \in k[Z]$ of smallest degree such that the operator $P(\phi^m)$ is right divisible by L in $k[\phi]$.

mHS₂: Constructing polynomials $P_0, P_1, \dots, P_{m-1} \in k[Z]$ such that if L has a solution in the form of an m -interlacing of some sequences $l^{(0)} = \langle l_n^{(0)} \rangle, l^{(1)} = \langle l_n^{(1)} \rangle, \dots, l^{(m-1)} = \langle l_n^{(m-1)} \rangle$, then $P_i(\phi)(l^{(i)}) = 0$:

$$P_i = \tau\phi^i P, \quad i = 0, 1, \dots, m-1.$$

mHS₃: Constructing finite sets $\mathcal{G}_i \subset k^*$, $i = 0, 1, \dots, m-1$, such that $V_1(P_i(\phi)) \subset V(\text{LCLM}_{h \in \mathcal{G}_i}(\phi - h))$ (one can use algorithms from [8], [6], and [3] for this).

mHS₄: Constructing an operator L_m such that $V_m(L) \subset V_m(L_m) = V(L_m)$:

$$L_m = \text{LCLM}_{h \in \mathcal{H}}(\phi^m - h), \quad (3)$$

where

$$\mathcal{H} = \bigcup_{0 \leq i \leq m-1} \{\phi^{-i}\tau^{-1}(h) \mid h \in \mathcal{G}_i\}. \quad (4)$$

mHS₅: Constructing the operator $H = \text{GCRD}(L, L_m)$ and a basis for $V(H)$ such that each element of this basis is the m -interlacing of hypergeometric sequences.

2.2 A Simplification of the Hendriks–Singer Procedure

The procedure mHS can be simplified by removing some unnecessary actions.

Theorem 1. *Let $\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_{m-1}$ be as in mHS₃, and $\mathcal{G} = \tau^{-1}\mathcal{G}_0$. In this case*

- (i) $\mathcal{G}_i = \tau\phi^i\tau^{-1}\mathcal{G}_0 = \tau\phi^i\mathcal{G}$, $i = 0, 1, \dots, m-1$;
- (ii) *one can use \mathcal{G} instead of \mathcal{H} in the right-hand side of [\(3\)](#).*

Proof. (i) For P as in mHS₁, and P_0, P_1, \dots, P_{m-1} as in mHS₂ we have

$$P_i = \tau\phi^i P = \tau\phi^i\tau^{-1}(\tau P) = \tau\phi^i\tau^{-1}P_0.$$

The proof follows from the definition of \mathcal{G} . (Notice that $\tau\phi^i\tau^{-1}$ is defined by $x \mapsto x + \frac{i}{m}$, and $\tau\phi^i$ is defined by $x \mapsto mx + i$.)

- (ii) We have $\mathcal{G}_i = \tau\phi^i\mathcal{G}$. Therefore, if $h \in \mathcal{G}_i$, then $\phi^{-i}\tau^{-1}(h) \in \mathcal{G}$. □

As a consequence of this theorem we obtain a simplified version of mHS which we denote by mHS':

mHS'₁: The same as mHS₁.

mHS'₂: Constructing the polynomial $P_0 = \tau P$.

mHS'₃: Constructing $\mathcal{G} = \tau^{-1}\mathcal{G}_0$, where the finite set $\mathcal{G}_0 \subset k^*$ is such that $V_1(P_0(\phi)) \subset V(\text{LCLM}(\phi - h))$.

mHS'₄: Set $L_m = \text{LCLM}_{h \in \mathcal{G}}(\phi^m - h)$.

mHS'₅: The same as mHS₅.

The cost of mHS'₁, mHS'₅ is the same as the cost of mHS₁, mHS₅. The cost of mHS'₂, mHS'₃, mHS'₄ is m times less than the cost of mHS₂, mHS₃, mHS₄.

Example 1. *Let*

$$L = \phi^5 - \frac{2}{x+5}\phi^4 + \frac{x-1}{x+5}\phi - \frac{2}{x+5}, \quad (5)$$

$m = 2$. *Then*

$$P(Z) = (x+10)(x+7)Z^5 - (4x+30)Z^4 + 4Z^3 - (x+4)(x+1)Z^2 + (4x+6)Z - 4,$$

$$P_0(\phi) = (2x+10)(2x+7)\phi^5 - (8x+30)\phi^4 + 4\phi^3 - (2x+4)(2x+1)\phi^2 + (8x+6)\phi - 4,$$

$$\mathcal{G}_0 = \left\{ \frac{1}{x+1}, \frac{2}{2x-1} \right\},$$

$$\mathcal{G} = \left\{ \frac{2}{x+2}, \frac{2}{x-1} \right\},$$

$$L_2 = \phi^4 - \frac{(4x+6)}{(x+4)(x+1)}\phi^2 + \frac{(4x+16)}{(x+1)},$$

$$H = \phi^2 + \frac{(12x+12)}{(x+2)(x^3-x-8)}\phi - \frac{(2x^3+6x^2+4x-16)}{(x+2)(x^3-x-8)}.$$

A basis for $V(H) = V_2(L)$ consists of two following sequences

the 2-interlacing of the sequences $\left\langle \frac{1}{\Gamma(n-1/2)} \right\rangle$ and $\left\langle \frac{1}{\Gamma(n+3/2)} \right\rangle$,

the 2-interlacing of the sequences $\left\langle \frac{1}{\Gamma(n+1)} \right\rangle$ and $\left\langle \frac{1}{\Gamma(n)} \right\rangle$.

Notice that once the set \mathcal{G} is constructed the operators L_m and H are not needed for constructing a basis for $V_m(L)$. This would simplify mHS'. But the operator H is used by the Hendriks–Singer algorithm for a recursion to construct all Liouvillian solutions of L (Section 5.3).

3 Some Special Cases

3.1 When \mathcal{C} Is Not Algebraically Closed

Suppose that \mathcal{C} is not algebraically closed. Then L may have hypergeometric solutions whose certificates belong to $\tilde{\mathcal{C}}(x)$ but not to $k = \mathcal{C}(x)$. However, the following statement holds (has been proven in [7]):

Let $L \in k[\phi]$ and each of the sets \mathcal{G}_i , $i = 0, 1, \dots, m$, constructed at the step mHS₃ contains all belonging to $\tilde{\mathcal{C}}(x)$ certificates of hypergeometric solutions of $P_i(\phi)$. Then the operator H computed at the step mHS₅ belongs to $k[\phi]$.

As a consequence we have that if $L \in k[\phi]$, and and the step mHS₄ we use some algorithm A for finding all certificates belonging to $\tilde{\mathcal{C}}(x)$, then we obtain $H \in k[\phi]$. The operator L is right-divisible by H , and we have $L = \tilde{L}H$, $\tilde{L} \in k[\phi]$. Even if the algorithm A is applicable only to operators from $k[\phi]$ then we always can apply this algorithm to \tilde{L} . The same is correct if we use mHS' instead of mHS for constructing H since we construct the same H in both cases. This fact might be quite important for finding all Liouvillian solutions of L , if the corresponding implementation of an algorithm for finding hypergeometric solution is not applicable to operators with the coefficients from $\tilde{\mathcal{C}}(x)$.

3.2 When L Is Irreducible

An algorithm which does not compute hypergeometric solutions of $P_0(\phi)$ was proposed in [2] for the case of an irreducible operator L . The idea of this algorithm is based on the notion of *gauge equivalence* of operators. Two operators $L, \tilde{L} \in k[\phi]$ are gauge equivalent if $\text{ord } L = \text{ord } \tilde{L}$ and there exists an operator T , $\text{ord } T < \text{ord } L$, such that $V(L) = T(V(\tilde{L}))$. If T exists then there also exists an “inverse” T' such that $V(\tilde{L}) = \tilde{T}'(V(L))$. An irreducible L is gauge equivalent to an operator of the form (6) iff L has Liouvillian solutions. When L is irreducible, getting (6) is equivalent to computing Liouvillian solutions (5). It was shown in [2] that this approach is very productive for irreducible L of order 2 or 3. However the gauge equivalence is not sufficient to get (6) for L in the general

case. The full factorization has a high complexity. In addition L may have a Liouvillian solution although $L = KM$ with an irreducible operator M which has no Liouvillian solution. It means one factorization $L = KM$ may not be sufficient for finding Liouvillian solution using factorization and paper [5]; it can happen that other factorizations $L = K'M'$ are needed to be searched for. So algorithms that are directly applicable in the general case are of definite value.

4 Some Properties of the Space $V_m(L)$

4.1 The Dimension of the Space $V_m(L)$

Lemma 1. *Any operator A of the form*

$$\phi^d - a(x), \quad a(x) \in k, \quad (6)$$

can be written as

$$A = \text{LCLM}(Q_1, Q_2, \dots, Q_l) \quad (7)$$

for some irreducible operators Q_1, Q_2, \dots, Q_l of the same order ρ , $l\rho = d$.

Proof. Let Q be any irreducible right factor of A : i.e. $A = BQ$ with an irreducible Q . Let R_k be the operator $\phi - e^{\frac{2\pi ki}{d}}$, and the sequences $z^{(k)} = \langle z_n^{(k)} \rangle$ be such that $z_n^{(k)} = e^{\frac{2\pi ki}{d}n}$, $k = 1, 2, \dots, d$ (recall that $k = \mathcal{C}(x)$ and the ground field \mathbb{C} is an algebraically closed subfield of \mathbb{C}). Set Q_k to be equal to the symmetric product of Q and R_k . The operator Q_k is monic irreducible and of same order as Q , $k = 1, 2, \dots, d$. Then $A = \text{LCLM}(Q_1, Q_2, \dots, Q_d)$ holds since for any $y \in V(A) \setminus \{0\}$ the sequences $z^{(k)}y$, $k = 1, 2, \dots, d$, are linearly independent elements of $V(A)$. Notice that some of operators Q_1, Q_2, \dots, Q_d can be equal. We get [7] with pairwise different irreducible Q_1, Q_2, \dots, Q_l after removing duplicates. \square

Lemma 2. *Let L be of the form [7], $V_j(L) = 0$ for $j = 0, 1, \dots, m-1$ and $V_m(L) \neq 0$. Let H be an operator such that $V(H) = V_m(L)$. Then H can be written as*

$$H = \text{LCLM}(S_1, S_2, \dots, S_t) \quad (8)$$

for some irreducible S_1, S_2, \dots, S_t of order m .

Proof. Follows from the construction of the operator H (see mHS₄, mHS₅), Lemma [1] and the fact that if H has a right factor of order $k < m$ then $V_j(L) \neq 0$ for some j such that $1 \leq j \leq k < m$. \square

Theorem 2. *Let L be of the form [7], $V_j(L) = 0$ for $j = 0, 1, \dots, m-1$ and $V_m(L) \neq 0$. Then m divides $\dim V_m(L)$.*

Proof. Follows from Lemma [2]. \square

¹ This proof is by M. van Hoeij (a private communication).

be the sources of the hypergeometric sequences (11). By Theorem 1(i) any element of $V_m(L)$ can be represented in the form

$$\left\langle \sum_{j=1}^s c_{0,j} U_j(n), \sum_{j=1}^s c_{1,j} U_j(n), \dots, \sum_{j=1}^s c_{m-1,j} U_j(n) \right\rangle, \quad (13)$$

with

$$U_j(x) = G_j \left(\frac{x}{m} \right), \quad j = 1, 2, \dots, s, \quad (14)$$

and with some concrete complex constants

$$c_{i,j}, \quad i = 0, 1, \dots, m-1, \quad j = 1, 2, \dots, s. \quad (15)$$

Note that in representation (13), all components of any element of $V_m(L)$ have identical structure, and each of the steps mHS_5 and mHS'_5 constructs a basis for the space of suitable constants (15).

Example 3. *The sequences belonging to the basis constructed in Example 1 can be presented as*

$$\left\langle \frac{c_1}{\Gamma(\frac{n}{2} - \frac{1}{2})} + \frac{c_2}{\Gamma(\frac{n}{2} + 1)}, \frac{c_2}{\Gamma(\frac{n}{2} - \frac{1}{2})} + \frac{c_1}{\Gamma(\frac{n}{2} + 1)} \right\rangle,$$

with $(1, 0), (0, 1)$ as (c_1, c_2) , and the sequences belonging to the basis constructed in Example 2 can be presented as

$$\left\langle c_1 (-2)^{n/2} \Gamma(n/2) + c_2 2^{n/2} \Gamma(n/2), c_3 (-2)^{n/2} \Gamma(n/2) + c_4 2^{n/2} \Gamma(n/2) \right\rangle$$

with $(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)$ as (c_1, c_2, c_3, c_4) .

We suppose that one unique value α of $2^{1/2}$ and, resp., one unique value β of $(-2)^{1/2}$ are selected. Then $2^{n/2} = \alpha^n$, $(-2)^{n/2} = \beta^n$. This is in agreement with the remark to definition of the source of a hypergeometric sequence.

The following proposition enables one to apply an operator of $k[\phi]$ to an m -interlacing of sequences.

Proposition 1. *If $\langle f_n^{(0)} \rangle, \langle f_n^{(1)} \rangle, \dots, \langle f_n^{(m-1)} \rangle$ are arbitrary sequences, then*

$$\phi \left(\left\langle f_n^{(0)}, f_n^{(1)}, \dots, f_n^{(m-1)} \right\rangle \right) = \left\langle f_{n+1}^{(1)}, f_{n+1}^{(2)}, \dots, f_{n+1}^{(m-1)}, f_{n+1}^{(0)} \right\rangle.$$

Proof. A direct check. □

4.3 Cyclic Permuted Solutions

Some of functions (12) can be *similar*, i.e., such that $G_i(x)/G_j(x) \in k$ for some indexes $i \neq j$ (the corresponding hypergeometric sequences are also called similar).

Lemma 3. Let $h = \langle h_n \rangle$ be a hypergeometric sequence with the source $G(x)$. Let $f = \langle f_n \rangle$ be the m -interlacing of sequences $\langle S_0(n)h_n \rangle, \langle S_1(n)h_n \rangle, \dots, \langle S_{m-1}(n)h_n \rangle$ with $S_0(x), S_1(x), \dots, S_{m-1}(x) \in k$. Then for some $R_0(x), R_1(x), \dots, R_m(x) \in k$ and $U(x) = G\left(\frac{x}{m}\right)$

$$f = \langle R_0(n)U(n), R_1(n)U(n-1), \dots, R_{m-1}(n)U(n-m+1) \rangle, \quad (16)$$

and

$$\begin{aligned} \phi(f) = \langle R_1(n+1)U(n), R_2(n+1)U(n-1), \dots \\ \dots, R_m(n+1)U(n-m+1) \rangle. \end{aligned} \quad (17)$$

Proof. Indeed, by definition of f

$$\begin{aligned} f = \left\langle S_0\left(\frac{n}{m}\right)G\left(\frac{n}{m}\right), S_1\left(\frac{n-1}{m}\right)G\left(\frac{n-1}{m}\right), \dots \right. \\ \left. \dots, S_{m-1}\left(\frac{n-m+1}{m}\right)G\left(\frac{n-m+1}{m}\right) \right\rangle. \end{aligned}$$

This proves (16). By Proposition 1 applying ϕ to a sequence of the form (16) gives

$$\begin{aligned} \langle R_1(n+1)U(n), R_2(n+1)U(n-1), \dots \\ \dots, R_{m-1}(n+1)U(n-m+2), R_0(n+1)U(n+1) \rangle. \end{aligned}$$

But $U(n+1) = U((n-m+1)+m) = S(n)U(n-m+1)$ with a rational function $S(x)$. Setting $R_{m+1}(x) = S(x-1)R_0(x)$ we get (17). \square

Lemma 4. Any element of $V_m(L)$ can be represented as a sum of solutions such that each of these solutions has the form of an m -interlacing of similar hypergeometric sequences.

Proof. We can present any element of $V_m(L)$ as a sum of the m -interlacings of similar hypergeometric sequences such that the components of different summands are not similar. Application of L to the m -interlacing of similar hypergeometric sequences gives again the m -interlacing of similar hypergeometric sequences whose components are similar to components of the original m -interlacing. The claimed follows. \square

Theorem 3. Let L have a solution (13). Then L has the cyclic permuted solution

$$\left\langle \sum_{j=1}^s c_{1,j} U_j(n), \sum_{j=1}^s c_{2,j} U_j(n), \dots, \sum_{j=1}^s c_{m-1,j} U_j(n), \sum_{j=1}^s c_{0,j} U_j(n) \right\rangle. \quad (18)$$

Proof. By Lemmas [3](#), [4](#) it is sufficient to consider the case [\(I6\)](#) of the m -interlacing of similar hypergeometric sequences. We have to prove that if [\(I6\)](#) is a solution of L then

$$\begin{aligned} & \langle R_1(n)U(n-1), R_2(n)U(n-2), \dots \\ & \dots, R_{m-1}(n)U(n-m+1), R_0(n)U(n) \rangle \end{aligned} \quad (19)$$

is also a solution of L . By the second part of Lemma [3](#) the result of applying L to [\(I9\)](#) has the form

$$\langle S_0(n)U(n-1), S_1(n)U(n-2), \dots, S_{m-1}(n)U(n-m) \rangle, \quad (20)$$

$S_0(x), S_1(x), \dots, S_{m-1}(x) \in k$.

We introduce the operation $'$:

$$\left\langle f_n^{(0)}, f_n^{(1)}, \dots, f_n^{(m-1)} \right\rangle' = \left\langle f_{n+1}^{(0)}, f_{n+1}^{(1)}, \dots, f_{n+1}^{(m-1)} \right\rangle$$

(in the case $m = 1$ this operation coincides with ϕ). For the operator [\(II\)](#) we set $L' = \phi^d + a_{d-1}(x+1)\phi^{d-1} + \dots + a_1(x+1)\phi + a_0(x+1)$. It is easy to see that $L'(f') = (L(f))'$ for any m -interlacing.

Using Proposition [II](#) we have

$$\begin{aligned} & L(\langle R_0(n)U(n), R_1(n)U(n-1), \dots, R_{m-1}(n)U(n-m+1) \rangle) = \\ & = L\phi^{-1}(\langle R_1(n+1)U(n), R_2(n+1)U(n-1), \dots \\ & \dots, R_{m-1}(n+1)U(n-m), R_0(n+1)U(n+1) \rangle) = \\ & = \phi^{-1}L'(\langle R_1(n+1)U(n), R_2(n+1)U(n-1), \dots \\ & \dots, R_{m-1}(n+1)U(n-m), R_0(n+1)U(n+1) \rangle) = \\ & = \phi^{-1}L'(\langle R_1(n)U(n-1), R_2(n)U(n-2), \dots \\ & \dots, R_{m-1}(n)U(n-m-1), R_0(n)U(n) \rangle)' = \\ & = \phi^{-1}((L(\langle R_1(n)U(n-1), R_2(n)U(n-2), \dots \\ & \dots, R_{m-1}(n)U(n-m-1), R_0(n)U(n) \rangle))' = \\ & = \phi^{-1}(\langle S_0(n+1)U(n), S_1(n+1)U(n-1), \dots \\ & \dots, S_{m-1}(n+1)U(n-m+1) \rangle) = \\ & = (\langle S_{m-1}(n)U(n-m), S_0(n)U(n-1), \dots, S_{m-2}(n)U(n-m+1) \rangle). \end{aligned}$$

Since [\(I6\)](#) is a solution of L we have

$$S_i(n) = 0 \quad \text{when} \quad n \equiv i + 1 \pmod{m}, \quad i = 0, 1, \dots, m-1.$$

But this implies that $S_i(x)$, $i = 0, 1, \dots, m-1$, are equal to zero identically. Therefore, [\(I9\)](#) is a solution of L . \square

Example 4. *The sequences belonging to the basis constructed in Example [1](#) can be presented as*

$$\left\langle \frac{1}{\Gamma(\frac{n}{2} - \frac{1}{2})}, \frac{1}{\Gamma(\frac{n}{2} + 1)} \right\rangle, \left\langle \frac{1}{\Gamma(\frac{n}{2} + 1)}, \frac{1}{\Gamma(\frac{n}{2} - \frac{1}{2})} \right\rangle.$$

5 Implementation

The improvements proposed in Section 2 are implemented as a modification of the MAPLE implementation [7] of the original Hendriks–Singer algorithm for finding Liouvillian solutions. The implementation is done as `LiouvillianSolution` function that extends the MAPLE package `LREtools` containing various functions for solving linear recurrence equations. To the best of the authors' knowledge it is the only one existing full implementations (at least, in MAPLE) of this algorithm.

The paper [1] presents a modification of the Hendriks–Singer algorithm HS and describes its implementation. However, the implementation described in that paper is not full since it solves the second-order equations only.

5.1 Liouvillian Solutions

By [5] the ring \mathcal{L} of Liouvillian sequences is the smallest subring of \mathcal{S} such that

$$k \subset \mathcal{L},$$

$$u \in \mathcal{L} \text{ iff } \phi(u) \in \mathcal{L},$$

$$u \in k \text{ implies that } v \in \mathcal{L} \text{ if } \phi(v) = uv,$$

$$u \in \mathcal{L} \text{ implies that } v \in \mathcal{L} \text{ if } \phi(v) = u + v,$$

$u^{(0)}, u^{(1)}, \dots, u^{(m-1)} \in \mathcal{L}$ implies that the interlacing of these sequences belongs to \mathcal{L} .

Note that the definition of Liouvillian sequences may be given in a different way, but the defined object is still the same. For example, by [9] (also used in [7]) the ring \mathcal{L} of Liouvillian sequences is the smallest subring of \mathcal{S} that contains the set of all hypergeometric elements from \mathcal{S} and closed with respect to ϕ , ϕ^{-1} , Σ (the summation), and the interlacing.

The space of all Liouvillian solutions of L will be denoted by $V_{\mathcal{L}}(L)$. We also will use notations $V(L), V_m(L)$ introduced before.

5.2 Finding the Operator H and a Basis for the Space $V_m(H)$ for a Fixed m

The implementation [7] was adjusted to utilize the procedure `mHS'` described in Section 2.2. In the initial implementation [7], the procedure `mHS` was an internal integral part of the function `LiouvillianSolution` to find all Liouvillian solutions. The output option `output=interlacing[m]` is added to `LiouvillianSolution` to provide users with a possibility to search for the space $V_m(L)$ for a given m . Note that m can be omitted in the option, and in this case, the function will find itself the smallest m for which $V_m(L) \neq 0$ (if such integer m exist). This is exactly what is needed as a first step for finding all Liouvillian solutions by HS, i.e., to construct the space $V_{\mathcal{L}}(L)$. Our implementation represents m -interlacing solutions in the form (13) using MAPLE's structure `piecewise` in the sense of the expression (10).

Example 5. Consider equation (5) from Example 7:

$$> \text{rec} := (n+5)*y(n+5)-2*y(n+3)+(n-1)*y(n+2)-2*y(n) :$$

There is no 1-interlacing solution:

```
> sol1 := LiouvillianSolution(rec,y(n),{}),
output=interlacing[1]);
```

sol1 := FAIL

There are 2-interlacing solutions as was presented in Example 4, and 2 is the smallest m for which there are m -interlacing solutions:

```
> sol2 := LiouvillianSolution(rec,y(n),{ },output=interlacing);
```

$$sol2 := \begin{cases} \frac{-C_1}{\Gamma(\frac{n}{2} + 1)} + \frac{-C_2}{\Gamma(\frac{n}{2} - \frac{1}{2})} & \text{irem}(n, 2) = 1 \\ \frac{-C_2}{\Gamma(\frac{n}{2} + 1)} + \frac{-C_1}{\Gamma(\frac{n}{2} - \frac{1}{2})} & \text{otherwise} \end{cases}$$

Now try to find 4-interlacing solutions:

```
> sol4 := LiouvillianSolution(rec,y(n),{ },
output=interlacing[4]);
```

$$sol4 := \begin{cases} \frac{\left(\frac{1}{4}\right)^{\left(\frac{n}{4}\right)} - C_1}{\Gamma\left(\frac{n}{4} + \frac{1}{4}\right)\Gamma\left(\frac{n}{4} - \frac{1}{4}\right)} + \frac{1}{4} \frac{\left(\frac{1}{4}\right)^{\left(\frac{n}{4}\right)} \sqrt{2} - C_2}{\Gamma\left(\frac{n}{4} + \frac{1}{2}\right)\Gamma\left(\frac{n}{4} + 1\right)} & \text{irem}(n, 4) = 1 \\ \frac{\left(\frac{1}{4}\right)^{\left(\frac{n}{4}\right)} - C_2}{\Gamma\left(\frac{n}{4} + \frac{1}{4}\right)\Gamma\left(\frac{n}{4} - \frac{1}{4}\right)} + \frac{1}{4} \frac{\left(\frac{1}{4}\right)^{\left(\frac{n}{4}\right)} \sqrt{2} - C_1}{\Gamma\left(\frac{n}{4} + \frac{1}{2}\right)\Gamma\left(\frac{n}{4} + 1\right)} & \text{irem}(n, 4) = 2 \\ \frac{\left(\frac{1}{4}\right)^{\left(\frac{n}{4}\right)} - C_1}{\Gamma\left(\frac{n}{4} + \frac{1}{4}\right)\Gamma\left(\frac{n}{4} - \frac{1}{4}\right)} + \frac{1}{4} \frac{\left(\frac{1}{4}\right)^{\left(\frac{n}{4}\right)} \sqrt{2} - C_2}{\Gamma\left(\frac{n}{4} + \frac{1}{2}\right)\Gamma\left(\frac{n}{4} + 1\right)} & \text{irem}(n, 4) = 3 \\ \frac{\left(\frac{1}{4}\right)^{\left(\frac{n}{4}\right)} - C_2}{\Gamma\left(\frac{n}{4} + \frac{1}{4}\right)\Gamma\left(\frac{n}{4} - \frac{1}{4}\right)} + \frac{1}{4} \frac{\left(\frac{1}{4}\right)^{\left(\frac{n}{4}\right)} \sqrt{2} - C_1}{\Gamma\left(\frac{n}{4} + \frac{1}{2}\right)\Gamma\left(\frac{n}{4} + 1\right)} & \text{otherwise} \end{cases}$$

They exist. But in this case, they actually correspond to the 2-interlacing solutions up to arbitrary constants transformation, which can be checked directly.

There is also an implementation consideration which does not relate to the algorithm efficiency but to the efficiency of the implementation in MAPLE. As follows from Theorem 1(ii) the elements of the union in the right-hand side of (4) are equal for all i . But if LCLM function is applied to \mathcal{H} directly without removing duplicated elements it works very ineffective. Since it is just a peculiarity of the MAPLE implementation rather than algorithm's feature, in order to check only the gain from the algorithm simplification itself, the trick of removing duplicates in \mathcal{H} before application of LCLM was added into our mHS implementation.

Example 6. Consider the following equation:

```
> rec := m^2*y(n+4+m)-((n+4)^2-m^2)*y(n+4)+(n-10)*m^2*y(n+1+m)
> -(n-10)*((n+1)^2-m^2)*y(n+1)-m^2*y(n+m)+(n^2-m^2)*y(n);
```

$$\begin{aligned} \text{rec} := & m^2 y(n+4+m) - ((n+4)^2 - m^2) y(n+4) + (n-10) m^2 y(n+1+m) \\ & - (n-10) ((n+1)^2 - m^2) y(n+1) - m^2 y(n+m) + (n^2 - m^2) y(n) \end{aligned}$$

The equation has m -interlacing solutions with the components which differ by constant factors. For example, let $m = 3$.

```
> m := 3: LiouvillianSolution(rec, y(n), {}),
> output=interlacing[m]);
```

$$\begin{cases} \Gamma\left(\frac{n}{3} + 1\right) \Gamma\left(-1 + \frac{n}{3}\right) \cdot C_1 & \text{irem}(n, 3) = 1 \\ \Gamma\left(\frac{n}{3} + 1\right) \Gamma\left(-1 + \frac{n}{3}\right) \cdot C_2 & \text{irem}(n, 3) = 2 \\ \Gamma\left(\frac{n}{3} + 1\right) \Gamma\left(-1 + \frac{n}{3}\right) \cdot C_3 & \text{otherwise} \end{cases}$$

To check the performance changes we use $m = 10$.

```
> m := 10:
```

Let us find 10-interlacing solutions (not printed, but it has the same structure as above for the case $m = 3$) and check the time needed to compute the result.

```
> st:=time():
> LiouvillianSolution(rec, y(n), {}), output=interlacing[m]:
> time()-st;
```

```
m=10:
```

```
---Finding P took 0.6 seconds
```

```
---Constructing L_10 took 3.3 seconds
```

```
---Computing H took 0.0 seconds
```

```
---Constructing basis took 0.1 seconds
```

```
4.063
```

```
> st:=time():
> LiouvillianSolution_old(rec,y(n),{}),output=interlacing[m]):
> time()-st;
```

```
m=10:
```

```
---Finding P took 0.6 seconds
```

```
---Constructing L_10 took 34.9 seconds
```

```
---Computing H took 0.0 seconds
```

```
---Constructing basis took 0.1 seconds
```

```
35.562
```

The old version (mHS) took more time than the simplified one (mHS') to check existence of 10-interlacing solution. But if we missed LCLM trick the old version is even worse. Note that unchanged parts in both versions took the same time.

```
> st:=time():
> LiouvillianSolution_old_no_trick(rec, y(n), {}),
> output=interlacing[m]):
> time()-st;

m=10:

---Finding P took 0.6 seconds

---Constructing L_10 took 74.5 seconds

---Computing H took 0.0 seconds

---Constructing basis took 0.1 seconds
75.125
```

5.3 Finding All Liouvillian Solutions

Liouvillian solutions of general form are constructed recursively by HS. The recursive application of mHS or mHS' leads to a factorization $L = RH_t \dots H_2 H_1$ where the operator R is such that $V_m(R) = 0$ for all integer $m \geq 1$, and where each of the operators H_i satisfies $V(H_i) = V_{m_i}(H_i) \neq 0$ for an integer $m_i \geq 1$. For any $i = 1, 2, \dots, t$ a basis B_i for $V_{m_i}(H_i)$ has to be constructed. Once a basis B_i for $V(H_i) = V_{m_i}(H_i)$ is constructed, $i = 1, 2, \dots, t$, algorithm HS constructs a basis B of $V(H_t \dots H_2 H_1) = V_{\mathcal{L}}(H_t \dots H_2 H_1) = V_{\mathcal{L}}(L)$, using the difference version of the method of variation of parameters ([4]). To do this HS solves $t - 1$ linear algebraic systems whose determinants are shifted Casoratians which correspond to the bases B_1, B_2, \dots, B_{t-1} for solutions spaces of the operators H_1, H_2, \dots, H_{t-1} .

Recall that the bases B_1, B_2, \dots, B_t and the basis B consist of the elements of the ring \mathcal{S} of the germs of sequences. The problem of defining integer n_0 such that any of the germs from B is a sequence (in the usual meaning) that satisfies L for all $n \geq n_0$ looks like quite actual. We will describe below two rules following which a suitable n_0 can be computed. Our implementation provides users with such n_0 in addition to B .

We will suppose that all operators under consideration are of the form

$$\phi^s + r_{s-1}(x)\phi^{s-1} + \dots + r_0(x), \quad (21)$$

where $r_0(x), r_1(x), \dots, r_{s-1}(x) \in k$. For an integer l we set $\mathbb{N}_l = \{n \in \mathbb{N}, n \geq l\}$. The mentioned rules are as follows.

1) If a rational function $h(x)$ is defined and does not vanish on \mathbb{N}_l then a hypergeometric sequence with certificate $h(x)$ is defined and does not vanish on \mathbb{N}_l . Note that the Casoratian of a basis for the solutions space of (21) also represents a hypergeometric sequence with $h(x) = (-1)^s r_0(x)$ (4).

2) If operators L, \tilde{L}, H are such that $L = \tilde{L}H$ and we use the procedure described in [5, Lemma 5.4.1] for constructing a basis for $V(H)$ in the form of a finite set of linear combinations of some computed sequences, then elements of the basis sequences are defined on \mathbb{N}_l if initial computed sequences are defined on \mathbb{N}_l and all coefficients of L, \tilde{L}, H are defined on \mathbb{N}_l as well.

Our implementation computes the Casoratian as a hypergeometric sequence using its certificate. The computed result may differ from the Casoratian by a constant factor. It still leads to computing correct basis elements since the elements are also defined up to an arbitrary constant non-zero factor.

Example 7. Consider again equation (5) from Example 1:

> `rec := (n+5)*y(n+5)-2*y(n+3)+(n-1)*y(n+2)-2*y(n) :`

Find all Liouvillian solutions. We use the implicit output form, since the explicit forms are too huge. The computation time is also printed:

```
> st:=time():
> LiouvillianSolution(rec, y(n), {} ,
> output=implicit,usepiecewise=true);
> time()-st;
```

$$\begin{aligned} & \left[-C_1 \left(\left(\sum_{i1=2}^{n-1} \left(-\frac{B_{12}(i1+1) B_{21}(i1)}{D_1(i1)} \right) \right) B_{11}(n) + \right. \right. \\ & \left. \left(\sum_{i1=2}^{n-1} \frac{B_{11}(i1+1) B_{21}(i1)}{D_1(i1)} \right) B_{12}(n) \right) + \\ & -C_2 \left(\left(\sum_{i1=2}^{n-1} \left(-\frac{B_{12}(i1+1) B_{22}(i1)}{D_1(i1)} \right) \right) B_{11}(n) + \right. \\ & \left. \left(\sum_{i1=2}^{n-1} \frac{B_{11}(i1+1) B_{22}(i1)}{D_1(i1)} \right) B_{12}(n) \right) + \\ & -C_3 \left(\left(\sum_{i1=2}^{n-1} \left(-\frac{B_{12}(i1+1) B_{23}(i1)}{D_1(i1)} \right) \right) B_{11}(n) + \right. \\ & \left. \left(\sum_{i1=2}^{n-1} \frac{B_{11}(i1+1) B_{23}(i1)}{D_1(i1)} \right) B_{12}(n) \right) + -C_4 B_{11}(n) + -C_5 B_{12}(n), \end{aligned}$$

$$\left[\begin{array}{l}
B_{21}(n) = \frac{(-1)^n (6 + 3n + n^2)}{(n+2)(n^3 - n - 8)}, \\
B_{22}(n) = \frac{\left(\frac{1}{2} - \frac{1}{2} I \sqrt{3}\right)^n (-3 - \sqrt{3} I + n \sqrt{3} I + n^2)}{(n+2)(n^3 - n - 8)}, \\
B_{23}(n) = -\frac{\left(\frac{1}{2} + \frac{1}{2} I \sqrt{3}\right)^n (3 - \sqrt{3} I + n \sqrt{3} I - n^2)}{(n+2)(n^3 - n - 8)}, \\
B_{11}(n) = \begin{cases} \frac{1}{\Gamma\left(\frac{n}{2} + 1\right)} & \text{irem}(n, 2) = 1 \\ \frac{1}{\Gamma\left(\frac{n}{2} - \frac{1}{2}\right)} & \text{otherwise} \end{cases}, \\
B_{12}(n) = \begin{cases} \frac{1}{\Gamma\left(\frac{n}{2} - \frac{1}{2}\right)} & \text{irem}(n, 2) = 1 \\ \frac{1}{\Gamma\left(\frac{n}{2} + 1\right)} & \text{otherwise} \end{cases}, \\
D_1(n) = \frac{(-2)^{(n+1)} ((n+1)^3 - n - 9)}{\Gamma(n+3)} \end{array} \right], 2 \leq n$$

4.125

The solutions basis is formed from a 2-interlacing basis of 2 elements and a 1-interlacing basis of 3 elements. The corresponding shifted Casoratian $D_1(n)$ is also presented. The expression is applicable for $n \geq 2$.

Acknowledgement. The authors wish to express their thanks to M. van Hoeij for useful discussions.

References

1. Bomboy, R.: Liouvillian Solutions of Ordinary Linear Difference Equations. In: Proc. 5th Internat. Workshop on Comp. Algebra in Scientific Computing, pp. 17–28 (2002)
2. Cha, Y., van Hoeij, M.: Liouvillian Solutions of Irreducible Linear Difference Equations. In: Proc. ISSAC 2009 (2009)
3. Cluzeau, T., van Hoeij, M.: Hypergeometric Solutions of Linear Difference Equations. AAEC 17(2), 83–115 (2006)
4. Elaydi, S.N.: An Introduction to Difference Equations. Springer, New York (1999)

5. Hendriks, P.A., Singer, M.F.: Solving Difference Equations in Finite Terms. *J. Symb. Comput.* 27, 239–259 (1999)
6. van Hoeij, M.: Finite singularities and hypergeometric solutions of linear recurrence equations. *J. Pure Appl. Algebra* 139, 109–131 (1999)
7. Khmelnov, D.E.: Search for Liouvillian solutions of linear recurrence equations in the MAPLE computer algebra system. *Programming and Computing Software* 34(4), 204–209 (2008)
8. Petkovšek, M.: Hypergeometric solutions of linear recurrences with polynomial coefficients. *J. Symb. Comput.* 14, 243–264 (1992)
9. Petkovšek, M.: Symbolic computation with sequences. *Programming and Computer Software* 32(2), 65–70 (2006)
10. van der Put, M., Singer, M.F.: *Galois Theory of Difference Equations*. LNM, vol. 1666. Springer, Heidelberg (1997)

Parametric Analysis of Stability Conditions for a Satellite with Gyrodines*

Andrey V. Banshchikov

Institute for System Dynamics and Control Theory,
Siberian Branch of Russian Academy of Sciences,
P.O. Box 292, 134, Lermontov str., Irkutsk, 664033, Russia

Abstract. With the aid of software LinModel elaborated on the basis of the computer algebra system “Mathematica” we have conducted an analysis of dynamics for a mechanical system, which represents an unguided satellite with 3 gyrodines on a circular orbit. Modeling of systems (i.e., constructing nonlinear and linearized differential equations of motion in the Lagrange 2nd kind form), as well as investigation of the issues of stability and gyroscopic stabilization for eight steady motions obtained, have been conducted on a PC with the aid of symbolic or symbolic-numeric computations. The domains of stability and stabilization constructed are represented in either an analytical form or graphic form.

1 Introduction

Productivity of computer algebra systems and efficiency of their application have been growing substantially side by side with the increase in the time optimality of contemporary computers and the capacity of their RAM. This gives the possibility to efficiently process larger volumes of symbolic information. When it becomes problematic to conduct investigation exclusively in symbolic form, it is possible to proceed to attributing definite values to some (or all) parameters of the scrutinized problem, and hence to remove the problem of rounding errors. This allows the researcher to penetrate deeper into the qualitative nature of the processes and to proceed to the quantitative (numerical) assessment on the later stage of the investigation.

The objective of the work is analysis of dynamics and stability of motion for a well-known orbital system [1] with the use of the software LinModel [2] elaborated on the basis of the computer algebra system (CAS) “Mathematica” (license L3228-8720).

Until now, in some works, which are not numerous and related to dynamics of satellites with 2- or 3-degrees of freedom gyroscopes, only some of possible steady motions have been investigated. The problem of defining the total set of steady motions has been stated for a satellite carrying only one gyroscope. In several cases, such a problem has been stated for similar systems under some

* The work has been supported by RF President’s grant (Project SS–1676.2008.1).

substantial restrictions, e.g. the absence of moments of gravitation forces. In the set of the publications, one can find the works (see e.g. [3]), in which the gyrodamping is realized with the use of various laws of control of the gyrodines. Such gravitation orientation of satellite is considered to be active. Meanwhile, in the present paper, we consider passive (unguided) orientation with the use of three gyroscopes. Any works, which consider the issue of possible gyroscopic stabilization of unperturbed motions of a satellite with gyrodines, are not known to the author. The author has to emphasize that problems related to stabilization of definite mechanical systems under the effect of diverse nature forces are of great interest at present.

2 Description of the System of Bodies under Investigation

Let, according to the approximate statement, point O_1 (the system's mass center) move along a circular orbit of radius R with some velocity, which is constant in its value. Introduce an orbital coordinate system Σ^1 at point O_1 such that axis O_1x_1 is directed along the radius of the orbit, while axis O_1y_1 is directed along the tangent to the orbital plane in the direction of the vector of linear velocity; axis O_1z_1 is directed along the normal to the orbital plane. The orientation of the orbital coordinate system is described in the inertial coordinate system Σ^0 with the center in the attracting point \bar{O} . Let us attribute the number of body 1 to the orbital coordinate system. The position of the satellite B_2 is determined with respect to the orbital coordinate system. There are 3 gyrodines

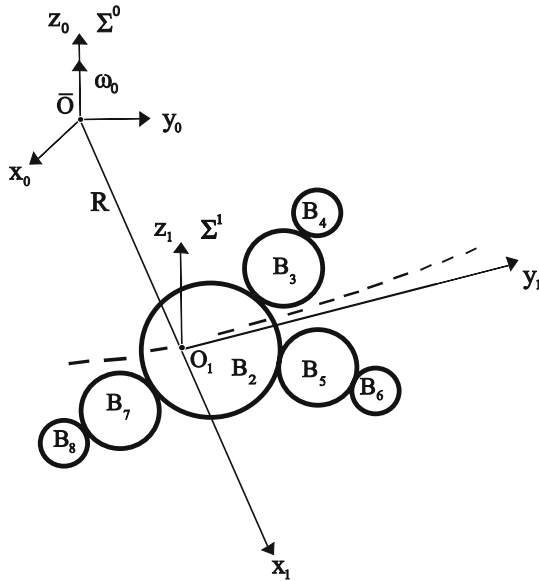


Fig. 1. The structure of interconnection between the bodies

installed on the satellite for the purpose of provision of proper orientation. The gyrodrine is a 2-degree-of-freedom system composed of a rotor and a frame. The rotor is fixed inside the frame and rotates with a constant angular rate. The frame and the rotor of each of gyrodrines are denoted in Fig. 1, resp., as bodies B_3, B_4 ; B_5, B_6 ; B_7, B_8 .

Consider initial data of the software LinModel, which assign geometric and kinematic characteristics of the satellite with its gyrodrines.

The number of bodies is 8.

Body B_1 (orbital coordinate system) is described with regard to the inertial coordinate system. The mass and the inertial tensor are zero. The numbers of the rotation axes are $\{3, 0, 0\}$, and the angles of rotations are $\{\chi, 0, 0\}$, i.e., the orbital coordinate system's orientation with respect to the inertial coordinate system is determined from one rotation at an angle χ about the 3rd axis Oz_0 . Zeros among the numbers of rotation axes indicate to the fact that there are no rotations about other axes. The radius vector of point O_1 in the coordinate system Σ^0 is $\mathbf{r}_{O_1}^0 = (R \cos \chi, R \sin \chi, 0)$. The radius vector of the mass center is $\mathbf{r}_{C_1}^1 = (0, 0, 0)$. The vector of absolute linear velocity of point O_1 is $\mathbf{V}_{O_1}^0 = (-R\omega_0 \sin \chi, R\omega_0 \cos \chi, 0)$, where $\omega_0 = \dot{\chi} = \text{const}$ is an angular rate of the orbital coordinate system.

Body B_2 (satellite) is described with regard to body 1. The body's mass is M_2 . Numbers of rotation axes are $\{1, 3, 2\}$, and the respective angles of rotations are $\{\psi, \theta, \varphi\}$, i.e., the orientation of the satellite with respect to the orbital coordinate system is determined by a sequence of three rotations: about the first axis O_1x_1 at an angle ψ ; about the third axis O_1z_1 at an angle θ ; about the second axis O_1y_1 at an angle φ . The radius vector of point O_2 in the orbital coordinate system is $\mathbf{r}_{O_2}^1 = (0, 0, 0)$, i.e., points O_1 and O_2 coincide. The radius vector of the mass center is $\mathbf{r}_{C_2}^2 = (0, 0, 0)$, i.e., the satellite's mass center is at point O_1 . The vector of relative linear velocity of point O_2 is $\mathbf{V}_{O_2}^1 = (0, 0, 0)$.

The body's tensor of inertia at point O_2 writes $\Theta_{O_2} = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix}$, i.e.,

J_x, J_y, J_z are satellite's main inertia moments.

Body B_3 (the first gyrodrine's frame) is connected with body 2. M_f is the mass of the frame, J_f is the inertia moment with respect to the axis of rotation of the frame itself (we neglect equatorial inertia moments). There takes place rotation of the frame about the 2nd axis at an angle α_1 . The radius vector of point O_3 , where the bodies connect with each other; its relative linear velocity; the radius vector of the mass center and the body's inertia tensor are, respectively,

$$\mathbf{r}_{O_3}^2 = \mathbf{V}_{O_3}^2 = \mathbf{r}_{C_3}^3 = (0, 0, 0); \quad \Theta_{O_3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & J_f & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Body B_4 (rotor of the first gyrodine) is connected with body 3. M_p is the mass of the rotor, J_p is the inertia moment with respect to the axis of its rotation (we neglect equatorial moments of inertia). There takes place rotation of the rotor about the first axis at an angle ϕ_1 . The radius vector of point O_4 , where the bodies are connected; its relative linear velocity; the radius vector of the mass center have (likewise for the previous body) zero components.

Pairs of bodies 5 – 6 and 7 – 8 represent, respectively, the 2nd and the 3rd gyrodines, whose geometric description and distribution of masses is similar to those of the 1st gyrodine's frame and rotor. The difference consists only in the axes proper rotations. As far as the 2nd gyrodine is concerned, rotation of the frame takes place about the 3rd axis at an angle of α_2 , and rotation of the rotor occurs about the 2nd axis at an angle ϕ_2 . As far as the 3rd gyrodine is concerned, rotation of the frame takes place about the 1st axis at an angle α_3 , and the rotation of the rotor occurs about the 3rd axis at an angle ϕ_3 .

So, the conservative mechanical system under scrutiny is described by the nine generalized coordinates $\varphi, \theta, \psi, \alpha_1, \phi_1, \alpha_2, \phi_2, \alpha_3, \phi_3$ and is located in the Newtonian field of gravitation to the center of circle \bar{O} .

3 Constructing a Symbolic Model

Let us list the set of problems solved in symbolic form with the aid of the software LinModel for the scrutinized system of bodies.

Problem 1. The following geometric and kinematic characteristics have been computed for each of the bodies:

- matrices of directional cosines;
- radius vectors of the mass centers and the points of connection of the bodies;
- relative and absolute angular rates of the bodies;
- relative and absolute linear velocities of the points of connection of the bodies.

Problem 2. The kinetic energy $T(\dot{\mathbf{q}}, \mathbf{q})$ for the system of bodies as a quadratic form with respect to generalized velocities $\dot{\mathbf{q}}$, and the force function $U(\mathbf{q})$ of the approximate Newtonian field of gravitation have been obtained. Here \mathbf{q} is the vector of generalized coordinates.

Problem 3. Nonlinear equations of motion in the Lagrange 2nd kind form have been constructed.

It has been determined that the three generalized coordinates ϕ_1, ϕ_2, ϕ_3 are cyclic, while the rest six coordinates are positional.

The formulas used in Problems 1 – 3 can be found, for example, in [4] or at the URL-address of [5].

Problem 4. Let us assign the unperturbed motion in the following form:

$$\begin{cases} \varphi = 0, \theta = 0, \psi = 0, \alpha_1 = \alpha_1^0, \alpha_2 = \alpha_2^0, \alpha_3 = \alpha_3^0, \dot{\varphi} = 0, \dot{\theta} = 0, \\ \dot{\psi} = 0, \dot{\alpha}_1 = 0, \dot{\alpha}_2 = 0, \dot{\alpha}_3 = 0, \dot{\phi}_1 = h_1, \dot{\phi}_2 = h_2, \dot{\phi}_3 = h_3. \end{cases} \quad (1)$$

Here h_1, h_2, h_3 are constant values for angular rates of proper rotations of the rotors; $\alpha_1^o, \alpha_2^o, \alpha_3^o$ are constant values of the angles of the gyrodine's frames. The given unperturbed motion is substituted into nonlinear equations of motion constructed at the previous step. Hence we obtain the following independent conditions of existence of motion (1):

$$\begin{cases} \cos \alpha_1^o J_p \omega_0 (h_1 - 4\omega_0 \sin \alpha_1^o) = 0, & \sin \alpha_3^o J_p \omega_0 (h_3 + \omega_0 \cos \alpha_3^o) = 0, \\ 3 \cos \alpha_2^o \sin \alpha_2^o J_p \omega_0^2 = 0, & -J_p \omega_0 \sin \alpha_2^o h_2 = 0, \quad -J_p \omega_0 \cos \alpha_2^o h_2 = 0. \end{cases} \quad (2)$$

Assuming that $\alpha_1^o, \alpha_2^o, \alpha_3^o$ have the values within the interval $[0, \pi/2]$, from equations (1) and (2) one can easily obtain the following eight solutions:

$$\begin{aligned} \varphi = 0, \quad \theta = 0, \quad \psi = 0, \quad \alpha_1 = \frac{\pi}{2}, \quad \alpha_2 = \frac{\pi}{2}, \quad \alpha_3 = 0, \\ \dot{\phi}_1 = h_1, \quad \dot{\phi}_2 = 0, \quad \dot{\phi}_3 = h_3; \end{aligned} \quad (3)$$

$$\begin{aligned} \varphi = 0, \quad \theta = 0, \quad \psi = 0, \quad \alpha_1 = \frac{\pi}{2}, \quad \alpha_2 = 0, \quad \alpha_3 = 0, \\ \dot{\phi}_1 = h_1, \quad \dot{\phi}_2 = 0, \quad \dot{\phi}_3 = h_3; \end{aligned} \quad (4)$$

$$\begin{aligned} \varphi = 0, \quad \theta = 0, \quad \psi = 0, \quad \alpha_1 = 0, \quad \alpha_2 = 0, \quad \alpha_3 = 0, \\ \dot{\phi}_1 = 0, \quad \dot{\phi}_2 = 0, \quad \dot{\phi}_3 = h_3; \end{aligned} \quad (5)$$

$$\begin{aligned} \varphi = 0, \quad \theta = 0, \quad \psi = 0, \quad \alpha_1 = \frac{\pi}{2}, \quad \alpha_2 = \frac{\pi}{2}, \quad \alpha_3 = \frac{\pi}{2}, \\ \dot{\phi}_1 = h_1, \quad \dot{\phi}_2 = 0, \quad \dot{\phi}_3 = 0; \end{aligned} \quad (6)$$

$$\begin{aligned} \varphi = 0, \quad \theta = 0, \quad \psi = 0, \quad \alpha_1 = \frac{\pi}{2}, \quad \alpha_2 = 0, \quad \alpha_3 = \frac{\pi}{2}, \\ \dot{\phi}_1 = h_1, \quad \dot{\phi}_2 = 0, \quad \dot{\phi}_3 = 0; \end{aligned} \quad (7)$$

$$\begin{aligned} \varphi = 0, \quad \theta = 0, \quad \psi = 0, \quad \alpha_1 = 0, \quad \alpha_2 = 0, \quad \alpha_3 = \frac{\pi}{2}, \\ \dot{\phi}_1 = 0, \quad \dot{\phi}_2 = 0, \quad \dot{\phi}_3 = 0; \end{aligned} \quad (8)$$

$$\begin{aligned} \varphi = 0, \quad \theta = 0, \quad \psi = 0, \quad \alpha_1 = 0, \quad \alpha_2 = \frac{\pi}{2}, \quad \alpha_3 = 0, \\ \dot{\phi}_1 = 0, \quad \dot{\phi}_2 = 0, \quad \dot{\phi}_3 = h_3; \end{aligned} \quad (9)$$

$$\begin{aligned} \varphi = 0, \quad \theta = 0, \quad \psi = 0, \quad \alpha_1 = 0, \quad \alpha_2 = \frac{\pi}{2}, \quad \alpha_3 = \frac{\pi}{2}, \\ \dot{\phi}_1 = 0, \quad \dot{\phi}_2 = 0, \quad \dot{\phi}_3 = 0. \end{aligned} \quad (10)$$

Definition. Above particular solutions of nonlinear equations of motion shall be called *steady motions*.

Note, for all the solutions (3) – (10), the second gyrodine's rotor must be at rest (i.e. $h_2 = 0$). Among the eight solutions there are two steady motions (3), (4) with parameters h_1, h_3 and two equilibrium positions (8), (10) without any parameters. The rest of the solutions have only one nonzero parameter (h_1 or h_3).

4 Stability Analysis

Algorithms intended for investigation (in symbolic-numerical form) of stability and stabilization of linearized models of mechanical systems can be found in [6], [7]. Consideration of above issues often leads to the problem of “parametric analysis” of algebraic inequalities obtained.

Example 1 (Steady motion (4)). Consider the problem of stability and of gyroscopic stabilization of solution (4).

Let us expand the Lagrangian $L = T(\dot{\mathbf{q}}, \mathbf{q}) + U(\mathbf{q})$ in series up to the 2nd degree of smallness with respect to the powers of deviations from the unperturbed motion (4). As a result, we construct equations of perturbed motion in the first approximation.

With the use of equations with respect to cyclic coordinates we exclude the cyclic velocities and accelerations from the obtained linear equations of motion. Hence we obtain an already “reduced” system of equations of the perturbed motion.

Let us present these equations in their explicit form, which coincides with their form of representation (a file of Notebook’s of CAS “Mathematica” format) in the software LinModel:

$$\left\{ \begin{array}{l} J_f(\ddot{\varphi} + \ddot{\alpha}_1) + (J_p(h_1 - \omega_0) + J_f\omega_0)\dot{\psi} + J_p(\varphi + \alpha_1)\omega_0(4\omega_0 - h_1) = 0, \\ J_f(\ddot{\theta} + \ddot{\alpha}_2) + 3J_p(\theta + \alpha_2)\omega_0^2 = 0, \\ J_f(\ddot{\psi} + \ddot{\alpha}_3) + (J_p(h_3 + \omega_0) - J_f\omega_0)\dot{\varphi} + J_p(\psi + \alpha_3)\omega_0(h_3 + \omega_0) = 0, \\ (J_f + J_y)\ddot{\varphi} + J_f\ddot{\alpha}_1 + ((J_f - J_p)\omega_0 - h_3J_p)\dot{\alpha}_3 \\ + (J_p(h_1 - h_3) + (J_f - 2J_p + J_x + J_y - J_z)\omega_0)\dot{\psi} \\ + \omega_0(\alpha_1J_p(4\omega_0 - h_1) + \varphi(J_p(h_3 - h_1) + 4(2J_p - J_x + J_z)\omega_0) = 0, \\ (J_f + J_x)\ddot{\psi} + J_f\ddot{\alpha}_3 + ((J_p - J_f)\omega_0 - J_ph_1)\dot{\alpha}_1 \\ + (J_p(h_3 - h_1) - (J_f - 2J_p + J_x + J_y - J_z)\omega_0)\dot{\varphi} \\ + \omega_0(\alpha_3J_p(h_3 + \omega_0) + \psi(J_p(h_3 - h_1) + (2J_p - J_y + J_z)\omega_0) = 0, \\ (J_f + J_z)\ddot{\theta} + J_f\ddot{\alpha}_2 + 3\omega_0^2((J_p - J_x + J_y)\theta + J_p\alpha_2) = 0. \end{array} \right. \quad (11)$$

From the coefficients of equations of system (11) we have constructed matrices A , G , C of the system’s characteristic equation: $|A\lambda^2 + G\lambda + C| = 0$, where A is a positive definite matrix of kinetic energy, and G , C are, respectively, matrices of gyroscopic and potential forces.

Kelvin–Chetayev’s theorems [8] allow one to start investigation of the issue of stability of the solution from analysis of the matrix of potential forces. The matrix of potential forces for solution (4) writes:

$$C = \begin{pmatrix} c_{11} & 0 & 0 & c_{14} & 0 & 0 \\ 0 & c_{22} & 0 & 0 & 0 & c_{26} \\ 0 & 0 & c_{33} & 0 & c_{35} & 0 \\ c_{14} & 0 & 0 & c_{44} & 0 & 0 \\ 0 & 0 & c_{35} & 0 & c_{55} & 0 \\ 0 & c_{26} & 0 & 0 & 0 & c_{66} \end{pmatrix}, \quad (12)$$

where $c_{22} = c_{26} = 3\omega_0^2 J_p$; $c_{44} = \omega_0(J_p(h_3 - h_1) + 4(J_z - J_x + 2J_p)\omega_0)$;
 $c_{55} = \omega_0(J_p(h_3 - h_1) + (J_z - J_y + 2J_p)\omega_0)$; $c_{33} = c_{35} = J_p \omega_0(h_3 + \omega_0)$;
 $c_{11} = c_{14} = J_p \omega_0(4\omega_0 - h_1)$; $c_{66} = 3\omega_0^2(J_p - J_x + J_y)$.

Let us write down the conditions of definite positiveness of matrix (12):

$$J_y > J_x, \quad 4\omega_0 - h_1 > 0, \quad h_3 + \omega_0 > 0, \quad h_1 + p_1\omega_0 < 0, \quad h_3 + p_2\omega_0 > 0, \quad (13)$$

where $p_1 = (J_y - J_z - J_p)/J_p$; $p_2 = 4(J_p - J_x + J_z)/J_p$.

Inequalities (13) are the sufficient conditions of stability of solution (4) with respect to positional coordinates.

Note that in practice absolute values of angular rates of the rotors are substantially in excess of ω_0 (i.e. $|h_1| \gg \omega_0$, $|h_3| \gg \omega_0$, and $0 < \omega_0 \ll 1$). Consequently, to satisfy conditions (13), rotors of the 1st and 3rd gyroindes shall rotate in different directions, i.e. $h_1 < 0$, $h_3 > 0$.

Suppose, some of conditions (13) are not satisfied (i.e., the potential system is unstable). Hence we may consider the issue of possibility of stabilization of the system's solution at the expense of influence of the gyroscopic forces.

It is known [8] that evenness (oddness) of the degree instability is determined by positiveness (negativity) of the determinant of matrix C of potential forces. It follows from Kelvin–Chetayev's theorem, which is related to the influence of the gyroscopic forces, that gyroscopic stabilization is possible only for those systems, which have an even degree of instability.

For example, if the second and fourth inequalities in (13) are replaced with inequalities of opposite sign $4\omega_0 - h_1 < 0$, $h_1 + p_1\omega_0 > 0$ then the determinant of matrix C from (12) remains positive, while some main diagonal minors of this matrix are negative. So, the system shall have an even degree of instability. Satisfaction of these two conditions necessitates that h_1 be positive.

The characteristic equation of system (11) $\Delta(\lambda^2) = \Delta_1(\lambda^2) * \Delta_2(\lambda^2) = 0$ contains λ only in the even powers. It is known that stability in such systems is possible only when all the roots of the polynomial Δ are purely imaginary, and, respectively, the roots with respect to λ^2 are negative and real numbers. This criterion [9] has been implemented in the form of program *CrKozlov* in the software *LinModel*.

Let us write down the algebraic conditions, which provide for existence of purely imaginary roots for the two co-factors $\Delta_1(\lambda^2)$, $\Delta_2(\lambda^2)$ of the characteristic equation of the “reduced” system of equations of motion.

The only condition $J_y > J_x$ provides the desired properties of the roots for the equation $\Delta_1(\lambda^2) = (J_f \lambda^2 + 3J_p \omega_0^2) (J_z \lambda^2 + 3(J_y - J_x)\omega_0^2) = 0$.

Coefficients of the equation $\Delta_2(\lambda^2) = a_0\lambda^8 + a_1\lambda^6 + a_2\lambda^4 + a_3\lambda^2 + a_4 = 0$ and the following necessary and sufficient conditions of existence of purely imaginary roots for the polynomial $\Delta_2(\lambda^2)$:

$$\left\{ \begin{array}{l} a_0 > 0, \quad a_1 > 0, \quad a_2 > 0, \quad a_3 > 0, \quad a_4 > 0, \quad 3a_1^2 - 8a_0a_2 > 0, \\ a_2^2a_1^2 - 3a_3a_1^3 - 6a_0a_4a_1^2 + 14a_0a_2a_3a_1 - 4a_0a_3^2 - 18a_0^2a_3^2 + 16a_0^2a_2a_4 > 0, \\ a_2^2a_3^2a_1^2 - 27a_2^2a_1^4 - 4a_3^3a_1^3 + 18a_2a_3a_4a_1^3 + 144a_0a_2a_4^2a_1^2 - 4a_2^3a_4a_1^2 \\ - 6a_0a_3^3a_4a_1^2 + 18a_0a_2a_3^3a_1 - 192a_0^2a_3a_4^2a_1 - 80a_0a_2^2a_3a_4a_1 - 27a_0^2a_3^4 \\ + 256a_0^3a_4^3 - 4a_0a_3^2a_3^2 - 128a_0^2a_2^2a_4^2 + 16a_0a_2^4a_4 + 144a_0^2a_2a_3^2a_4 > 0 \end{array} \right. \quad (14)$$

have a rather bulky analytical form, and are not given herein explicitly.

Note, $a_4 = \frac{\det C}{9J_p(J_y - J_x)\omega_0^4}$. Having written the analytical solution for the positiveness of coefficient a_4 , we obtain the three possible variants of values J_z :

$$0 < J_z \leq J_x - \frac{3J_p}{4}; \quad J_x - \frac{3J_p}{4} < J_z \leq 3J_p + J_y; \quad J_z > 3J_p + J_y. \quad (15)$$

Parametric analysis of inequalities (14) has been conducted in symbolic-numeric form. Furthermore, we have used the functions `Reduce`, `RegionPlot`, `RegionPlot3D` of CAS “Mathematica” intended, respectively, for solving the systems of algebraic inequalities as well as for obtaining the graphic 2D- and 3D-representations of solutions of these systems.

The following numerical values related to distribution of masses in the system have been introduced: $J_p = 3$, $J_f = 1$, $J_x = 230$, $J_y = 310$ (dimension of inertia moments: kg m^2), and $\omega_0 = 0.0011 \text{ rad c}^{-1}$. For each of three intervals (15) we have constructed graphic domains (in the space of parameters J_z, h_1, h_3), in which inequalities (14) are satisfied. It has been determined that the largest domain of gyroscopic stabilization shall take place for the first interval from (15).

For example, in Fig. 2 we give a solution of inequalities (14) for the third interval when $J_z = 350$. The dotted domains are the domains of gyroscopic stabilization for solution (4). It is obvious from the figure that – in order to provide for the system’s stabilization – the 3rd gyrodine’s rotor shall rotate more than six times as fast (or slow) as the 1st gyrodine’s rotor.

Example 2 (Steady motion (8)). Let us investigate the issue of stability of the equilibrium position (8).

It has been found out, there exists an additional fourth cyclic integral in the “reduced” system of equations of the perturbed motion in the first approximation (with respect to coordinate α_3) for solution (8). Respectively, the characteristic equation of the initial system has four zero roots of multiplicity two.

Stability analysis of solution (8) is conducted with respect to five positional coordinates $\alpha_1, \alpha_2, \varphi, \psi, \theta$ and four cyclic velocities $\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3, \dot{\alpha}_3$.

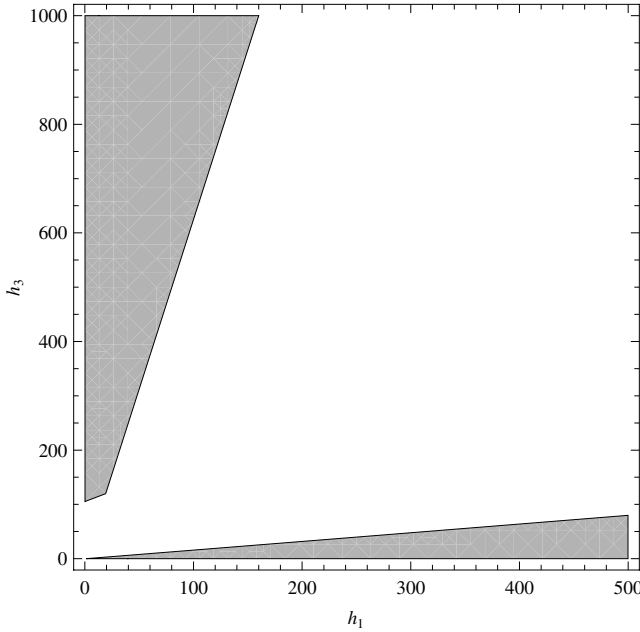


Fig. 2. The domain of gyroscopic stabilization

Having obtained equations of the perturbed motion in the neighborhood of solution (8), let us represent the matrix of potential forces already for the “reduced” system of equations:

$$C = \omega_0^2 \begin{pmatrix} -3J_p & 0 & -3J_p & 0 & 0 \\ 0 & 3J_p & 0 & 0 & 3J_p \\ -3J_p & 0 & J_f - 3J_p - 4J_x + 4J_z & 0 & 0 \\ 0 & 0 & 0 & J_z - J_y & 0 \\ 0 & 3J_p & 0 & 0 & 3(J_p - J_x + J_y) \end{pmatrix} \quad (16)$$

Matrix C from (16) is not definite positive, since some main diagonal minors are negative. But under the conditions $J_f - 4J_x + 4J_z > 0$, $J_y > J_x$, $J_y > J_z$, the determinant of this matrix is positive, and the system shall have an even degree of instability. In this case, let us consider the problem of possibility of gyroscopic stabilization.

The characteristic equation of the “reduced” system of equations of perturbed motion for the scrutinized solution has the form:

$$(J_f \lambda^2 + 3J_p \omega_0^2)(J_z \lambda^2 + 3(J_y - J_x) \omega_0^2)(a_0 \lambda^6 + a_1 \lambda^4 + a_2 \lambda^2 + a_3) = 0$$

$$\begin{aligned}
\text{where } a_0 &= J_f J_x J_y ; & a_3 &= 3J_p (J_y - J_z) (J_f - 4J_x + 4J_z) \omega_0^6 ; \\
a_1 &= \left((2(J_y + J_z) J_x + J_z (J_z - J_y) - 3J_x^2) J_f - 3J_p J_x J_y \right. \\
&\quad \left. - (J_x + J_y - 2J_z) J_f^2 + J_f^3 \right) \omega_0^2 ; \\
a_2 &= \left(3J_p (3J_x^2 - 2(J_y + J_z) J_x + (J_y - J_z) J_z) - (3J_p + 4J_x + J_y - 5J_z) J_f^2 \right. \\
&\quad \left. + J_f^3 + (3J_p (J_x + 2J_y - 2J_z) + 4(J_x - J_z) (J_y - J_z)) J_f \right) \omega_0^4 .
\end{aligned}$$

It is known that the necessary condition of stabilization is the positiveness of all the coefficients in the characteristic equation. With the use of above function **Reduce** we have managed to find out that these conditions are incompatible.

$$\begin{aligned}
\text{In}[1] &= \text{Reduce} [J_f > 0 \wedge J_f < J_p < J_x < J_z < J_y \wedge a_1/\omega_0^2 > 0 \\
&\quad \wedge a_2/\omega_0^4 > 0 \wedge a_3/\omega_0^6 > 0, \{J_f, J_p, J_x, J_z, J_y\}, \text{Reals}]
\end{aligned}$$

Out[1] = False

So, the simultaneous positiveness of coefficients a_1, a_2, a_3 cannot be provided. Consequently, gyroscopic stabilization of the equilibrium position (8) is impossible.

As a result of stability analysis of other steady motions we have obtained the following results:

a) Solutions (3), (6), (9), and (10) are unstable because one of the co-factors $\lambda^2 J_f - 3J_p \omega_0^2 = 0$ of the system's characteristic equation has a positive real root with respect to λ^2 . Note, for all these solutions, $\alpha_2^2 = \pi/2$.

b) Solution (5) is potentially unstable and cannot be stabilized at the expense of the effect of gyroscopic forces with an even degree of instability.

c) Solution (7) is potentially stable (with respect to five positional coordinates) in case of satisfaction of the obtained conditions of positive definiteness for the matrix of potential forces. If these conditions are not satisfied, then the issue of the possibility of gyroscopic stabilization for the system, which has an even degree of instability, is solved positively.

5 Conclusion

It is necessary to emphasize that problems of plausibility and precision of computations, as well as problems of explicitness and time optimality of the process of investigations can be partially solved, when a computer algebra system (CAS) is chosen in the capacity of the software tool. Side by side with employment of CAS (as the "computing tool") for solving a definite problem, the approach, which presumes elaboration of the software for solving a definite class of problems on the basis of the CAS's (in our case – CAS "Mathematica") internal programming language, may be considered as more valuable. Practically, above analysis on the whole has been conducted with the use of such software.

Summarizing the results, we have to note some provisions related to the timing data and the computational environment:

1) Execution of programs inside CAS “Mathematica” takes place only in the mode of interpretation, so, the total process of investigation with the aid of the software LinModel is conducted in the environment of CAS “Mathematica”.

2) The description of the mechanical system is formed in interactive mode, when the user inputs above data (see section 2) into the corresponding windows of the software. This important process is conducted just once. Later it is always possible to correct input data already in the created file when there is the need.

3) Modeling of the system (i.e., constructing nonlinear and linearized differential equations of motion in the neighborhood of one of eight steady motions) has necessitated some 100 Megabyte of RAM and 11 seconds of CPU time on a PC with the processor Core 2 Duo 2.40 GHz.

4) The process of investigation related to stability of steady motions has been conducted already not in automatic mode. At this stage, we have employed our own software, side by side with built-in functions and standard add-on packages of CAS “Mathematica”.

References

1. Sarychev, V.A.: Problems of orientation of satellites. In: Itogi Nauki i Tekhniki. Series “Space Research”. Moscow, VINITI Publ. 11 (1978) (in Russian)
2. Banshchikov, A.V., Burlakova, L.A., Irtegov, V.D., Titorenko, T.N.: A software LinModel intended for analysis of dynamics of large-dimensional mechanical systems. Certificate of State Registration of Computer Software No.2008610622. FGU FIPS, issued February 1 (2008) (in Russian)
3. Banshchikov, A.V., Bourlakova, L.A.: On Stability of a Satellite with Gyrodines. In: Proc. Seventh Workshop on Computer Algebra in Scientific Computing, pp. 61–69. Technische Universität München (2004)
4. Banshchikov, A.V., Burlakova, L.A., Irtegov, V.D., Titorenko, T.N.: The tasks of mechanics and computer algebra. *Mathematical Machines and Systems* 4, 82–97 (2008) (in Russian)
5. Banshchikov, A.V., Bourlakova, L.A., Irtegov, V.D., Ivanova, G.N., Titorenko, T.N.: Symbolic Computations in Problems of Mechanics. *Meixner Symbolic Computations in Problems of Mechanics*. In: Proc. Int. Mathematica Symposium. *The Mathematica Journal*, vol. 8(2) (2001), <http://south.rotol.ramk.fi/keranen/IMS99/paper15/ims99paper15.pdf>
6. Banshchikov, A.V., Bourlakova, L.A.: Information and Research System “Stability”. *Journal of Computer and Systems Sciences International* 35(2), 177–184 (1996)
7. Banshchikov, A.V., Bourlakova, L.A.: Computer algebra and problems of motion stability. *Mathematics and Computer in Simulation* 57, 161–174 (2001)
8. Chetayev, N.G.: *Stability of Motion*. Works on Analytical Mechanics. AS USSR, Moscow (1962) (in Russian)
9. Kozlov, V.V.: Stabilization of the unstable equilibria of charges by intense magnetic fields. *J. Appl. Math. and Mech.* 61(3), 377–384 (1997)

Computing and Visualizing Closure Objects Using Relation Algebra and RELVIEW

Rudolf Berghammer and Bernd Braßel

Institut für Informatik, Christian-Albrechts-Universität Kiel
Olshausenstraße 40, D-24098 Kiel
{rub, bbr}@informatik.uni-kiel.de

Abstract. Closure systems and closure operations play an important role in both mathematics and computer science. In addition there are a number of concepts which have been proven to be isomorphic to closure systems and we refer to all such concepts as *closure objects*. In this work we develop relation-algebraic specifications to recognize several classes of closure objects, compute the complete lattices they constitute and transform any of these closure objects into another. All specifications are algorithmic and can directly be translated into the programming language of the computer algebra system RELVIEW, which is a special purpose tool for computing with relations. We show that the system is well suited for computing and visualizing closure objects and their complete lattices.

1 Introduction

The procedure of computing the closure of a given object is an important basic technique for applications in mathematics and computer science alike. The approach followed most frequently is to employ so-called closure operations. Examples include operations that yield the transitive closure of a relation, the convex hull of a set of points in real vector spaces or the subgroup generated by a set of elements of a given group. The practical importance of closures is also documented by the fact that equivalent or at least very similar notions are frequently reinvented. For example, what [6] denotes as “full implicational system” is called a “full family of functional dependencies” in the theory of relational databases [7] or a “closed family of implications” in formal concept analysis [9]. Likewise a “dependency relation” [6, Section 2.2] is the same as a “contact relation” in the sense of [1] and one needs only to transpose such a relation and restrict its range to $2^X \setminus \{\emptyset\}$ in order to obtain an “entailment relation” in the sense of [8]. Moreover, on the lattice theoretic side it has been proven that there exists a close correspondence between all of the aforementioned concepts and the concept of a closure operation on a set. All of these objects form complete lattices which are pairwise isomorphic [6].

The subject of this work is to give a relation-algebraic representation of closure objects. The presented formulas are proven to be correct with respect to the

original formulation in predicate logic. The resulting specifications are algorithmic and directly lead to corresponding programs for RELVIEW [2,5], which is a computer algebra system for the special purpose of computing with relations. The developed formulas provide three basic algorithms for each closure object given as a finite relation, viz. *recognition of an object*, i.e., deciding whether a given relation is, for example, a dependency relation, *transformation between closure objects*, e.g., compute the closure operation corresponding to a given full implicational system, and *computation of the complete lattice* which is constituted by the set of all such closure objects. Employing the resulting programs the computer algebra system RELVIEW supports the study of specific closure objects in several ways, among which are *visualization* as Boolean matrices or as graphs, providing various graph layout algorithms and offering a number of ways to highlight selected portions, *retracing* the results of sub expressions via step-wise execution, and *testing*, e.g., by providing random matrices of a specified degree of filling. Beyond the possibility to study individual closure objects, the presented algorithms scale well and are applicable to large examples. This is due to the fact that finite relations can be implemented efficiently with ROBDDs [10] and we have taken care that the developed formulas fit the setting of RELVIEW. The only exception is the computation of all possible closure systems which is presented in Section 3.1 as the according problem is well known to be exponential for sets.

2 Relation Algebra

Since many years relation algebra in the sense of [12,11] is used by many mathematicians, computer scientists and engineers as conceptual and methodological base of their work. Its practical importance is due to the fact that many important objects of discrete mathematics can be seen as specific relations. Relation algebra allows very concise and exact problem specifications and, combined with predicate logic, extremely formal and precise calculations that drastically reduce the danger of making mistakes during the algorithm developments. In this section, we first recall the basics of relation algebra (Section 2.1), provide the reader with an introduction of how pairs and products can be modeled (Section 2.2) and then show how to represent sets (Section 2.3). More specific to the task at hand Section 2.4 deals with the relation-algebraic specification of extremal elements of ordered sets and lattices.

2.1 Relations and Relation Algebra

We write $R : X \leftrightarrow Y$ if R is a relation with domain X and range Y , i.e., a subset of the direct product $X \times Y$. If the sets X and Y of R 's *type* $[X \leftrightarrow Y]$ are finite and of size m and n , respectively, we may consider R as a Boolean matrix with m rows and n columns. Since the Boolean matrix interpretation is well suited for many purposes and is also used by the computer algebra system RELVIEW as one of its possibilities to depict relations, we often use matrix terminology and matrix

notation. Especially we speak about the rows, columns and entries of a relation and write $R_{x,y}$ instead of $\langle x, y \rangle \in R$ or $x R y$. Relation algebra knows three basic relations. The *identity relation* $I : X \leftrightarrow X$ satisfying for all $x, y \in X$ that $I_{x,y}$ iff $x = y$, the *universal relation* $L : X \leftrightarrow Y$ holding for all $x \in X$ and $y \in Y$, and the *empty relation* $O : X \leftrightarrow Y$ which holds for no pair in $X \times Y$. The *transposition* of a given relation $R : X \leftrightarrow Y$ is denoted by $R^\top : Y \leftrightarrow X$ and satisfies for all x, y that $R_{x,y}^\top$ iff $R_{y,x}$. Transposition allows, e.g., a compact formulation that a given relation R is symmetric by simply stating that it satisfies $R = R^\top$. When viewing relations as sets it comes natural to form the *union* $R \cup S : X \leftrightarrow Y$ and *intersection* $R \cap S : X \leftrightarrow Y$ of two relations $R, S : X \leftrightarrow Y$ or to state the *inclusion* $R \subseteq S$. The suggestive meaning is, respectively, that for all x, y we have $R_{x,y}$ or $S_{x,y}$ for $R \cup S$, both $R_{x,y}$ and $S_{x,y}$ for $R \cap S$ and that $R_{x,y}$ implies $S_{x,y}$ for $R \subseteq S$. The fact that, for example, a given relation R is reflexive can thus simply be expressed as $I \subseteq R$. A lot of the expressive power of relation algebra is due to the possibility to express the *composition* $RS : X \leftrightarrow Y$ of two relations $R : X \leftrightarrow Z$ and $S : Z \leftrightarrow Y$. Its definition in predicate logic is that for all $x \in X$ and $y \in Y$ we have $RS_{x,y}$ iff there exists a $z \in Z$ such that $R_{x,z}$ and $S_{z,y}$. Employing composition the fact that a given relation is transitive is concisely expressed by $RR \subseteq R$. The *complementation* of a relation $R : X \leftrightarrow Y$ is denoted by $\overline{R} : X \leftrightarrow Y$ and corresponds to negation in predicate logic: for all x, y we have $\overline{R}_{x,y}$ iff $R_{x,y}$ does not hold.

The *symmetric quotient* of two relations $R : X \leftrightarrow Y$ and $S : X \leftrightarrow Z$ is not a basic construct of relation algebra but can be defined by

$$\text{syq}(R, S) := \overline{R^\top S} \cap \overline{R}^\top S : Y \leftrightarrow Z.$$

The corresponding definition in predicate logic is that $\text{syq}(R, S)_{y,z}$ iff for all x we have that $R_{x,y}$ iff $S_{x,z}$. In other words for all $y \in Y$ and $z \in Z$ we have $\text{syq}(R, S)_{y,z}$ iff the y -column of R equals the z -column of S . Additional properties of this construct can be found in [11].

2.2 Pairing and Related Constructions

The *pairing* (or *fork*) $[R, S] : Z \leftrightarrow X \times Y$ of two relations $R : Z \leftrightarrow X$ and $S : Z \leftrightarrow Y$ is defined by demanding for all $z \in Z$ and $u = \langle u_1, u_2 \rangle \in X \times Y$ that $[R, S]_{z,u}$ iff R_{z,u_1} and S_{z,u_2} . (It should be noted that throughout this paper pairs $u \in X \times Y$ are assumed to be of the form $\langle u_1, u_2 \rangle$.) Using identity and universal relations of appropriate types, the pairing operation allows to define the two *projection relations* $\pi : X \times Y \leftrightarrow X$ and $\rho : X \times Y \leftrightarrow Y$ of the direct product $X \times Y$ as $\pi := [I, L]^\top$ and $\rho := [L, I]^\top$. Then the above definition implies for all $u \in X \times Y$, $x \in X$ and $y \in Y$ that $\pi_{u,x}$ iff $u_1 = x$ and $\rho_{u,y}$ iff $u_2 = y$. Also the *parallel composition* (or *product*) $R \parallel S : X \times X' \leftrightarrow Y \times Y'$ of two relations $R : X \leftrightarrow Y$ and $S : X' \leftrightarrow Y'$, such that $(R \parallel S)_{u,v}$ is equivalent to R_{u_1,v_1} and S_{u_2,v_2} for all $u \in X \times X'$ and $v \in Y \times Y'$, can be defined by means of pairing. We get the desired property if we define $R \parallel S := [\pi R, \rho S]$, where $\pi : X \times X' \leftrightarrow X$ and $\rho : X \times X' \leftrightarrow X'$ are the projection relations on $X \times X'$.

2.3 The Representation of Sets

There are several possibilities to model sets in relation algebra. Firstly, sets can be modeled using *vectors*, which are relations v which satisfy $v = vL$. For a vector the range is irrelevant and we therefore consider vectors $v : X \leftrightarrow 1$ with a specific singleton set $1 = \{\perp\}$ as range and omit 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 X . A non-empty vector v is said to be a *point* if $vv^T \subseteq 1$, i.e., v is *injective*. This means that it represents a singleton set and we frequently identify this singleton set with the only element it contains. In the Boolean matrix model a point $v : X \leftrightarrow 1$ is a Boolean column vector in which exactly one entry is 1.

As a second way to model sets we will apply the relation-level equivalents of the set-theoretic symbol \in , i.e., *membership-relations* $M : X \leftrightarrow 2^X$ on X and its powerset 2^X . These specific relations are defined by demanding for all $x \in X$ and $Y \in 2^X$ that $M_{x,Y}$ iff $x \in Y$. A Boolean matrix implementation of M requires exponential space. Using reduced ordered binary decision diagrams (ROBDDs) as implementation of relations (as in RELVIEW), however, the number of ROBDD-nodes for M is linear in the cardinality of X . See [10,4] for details.

Finally, we use injective functions to model sets. Given an injective function ι from Y to X , we may consider Y as a subset of X by identifying it with its image under ι . If Y is actually a subset of X and ι is given as relation of type $[Y \leftrightarrow X]$ such that $\iota_{y,x}$ iff $y = x$ for all $y \in Y$ and $x \in X$, then the vector $\iota^T L : X \leftrightarrow 1$ represents Y as subset of X in the sense above. Clearly, the transition in the other direction is also possible, i.e., the generation of a relation $\text{inj}(v) : Y \leftrightarrow X$ from the vector representation $v : X \leftrightarrow 1$ of $Y \subseteq X$ such that for all $y \in Y$ and $x \in X$ we have $\text{inj}(v)_{y,x}$ iff $y = x$.

A combination of injective functions with membership-relations allows a *column-wise enumeration* of sets of subsets. More specifically, if $v : 2^X \leftrightarrow 1$ represents a subset \mathfrak{S} of the powerset 2^X in the sense defined above, then for all $x \in X$ and $Y \in \mathfrak{S}$ we get the equivalence of $(M \text{inj}(v)^T)_{x,Y}$ and $x \in Y$. This means that $S := M \text{inj}(v)^T : X \leftrightarrow \mathfrak{S}$ is the relation-algebraic specification of membership on \mathfrak{S} , or, using matrix terminology, the elements of \mathfrak{S} are represented precisely by the columns of S . Furthermore, a little reflection shows for all $Y, Z \in \mathfrak{S}$ the equivalence of $Y \subseteq Z$ and $\overline{S^T \overline{S}}_{Y,Z}$. Therefore, $\overline{S^T \overline{S}} : \mathfrak{S} \leftrightarrow \mathfrak{S}$ is the relation-algebraic specification of set inclusion on \mathfrak{S} .

2.4 Extremal Elements of Orders and Lattices

Given a relation $R : X \leftrightarrow X$, the pair (X, R) is a partial order set iff $I \subseteq R$ (reflexivity), $R \cap R^T \subseteq I$ (antisymmetry) and $RR \subseteq R$ (transitivity) hold. In the following we may omit the set X when clear from context and simply refer to R as a partial order. When dealing with ordered sets, one typically investigates extremal elements. Based upon the vector representation of sets we will use the following relation-algebraic specifications taken from [11].

$$\begin{aligned}
 \text{lel}(R, v) &:= v \cap \overline{\overline{R}v} & \text{gel}(R, v) &:= \text{lel}(R^\top, v) \\
 \text{glb}(R, v) &:= \text{gel}(R, \overline{\overline{R}v}) & \text{lub}(R, v) &:= \text{glb}(R^\top, v)
 \end{aligned} \tag{1}$$

If $R : X \leftrightarrow X$ is a partial order relation and Y a subset of X that is represented by the vector $v : X \leftrightarrow 1$, then $\text{lel}(R, v) : X \leftrightarrow 1$ is empty iff Y does not have a least element and is a point that represents the least element of Y , otherwise. Similarly, $\text{gel}(R, v) : X \leftrightarrow 1$ ($\text{glb}(R, v) : X \leftrightarrow 1$ and $\text{lub}(R, v) : X \leftrightarrow 1$, respectively) is either empty or a point that represents the greatest element (greatest lower bound and least upper bound, respectively) of Y ,

If the second arguments of the specifications of [\(1\)](#) are not vectors but “proper” relations with a non-singleton range, then the corresponding extremal elements are computed column-wisely. E.g., in the case of the first specification this means the following. For all $A : X \leftrightarrow Y$ we obtain $\text{lel}(R, A) : X \leftrightarrow Y$ and, furthermore, for all $x \in X$ and $y \in Y$ that $\text{lel}(R, A)_{x,y}$ iff the least element of $\{z \in X \mid A_{z,y}\}$ exists and equals x . Hence, for all $y \in Y$ the y -column of $\text{lel}(R, A)$ is either empty or a point that represents (with respect to R) the least element of the set the y -column of A represents.

For a partial order $R : X \leftrightarrow X$ we also need the following specifications:

$$\text{Inf}(R) := [R, R]^\top \cap \overline{\overline{[R, R]^\top R}} \quad \text{Sup}(R) := \text{Inf}(R^\top) \tag{2}$$

Both specifications of [\(2\)](#) are of type $[X \times X \leftrightarrow X]$ and it is shown in [\[3\]](#) that for all $u \in X \times X$ and $x \in X$ we have $\text{Inf}(R)_{u,x}$ iff x is the greatest lower bound of u_1 and u_2 and $\text{Sup}(R)_{u,x}$ iff x is the least upper bound of u_1 and u_2 . Hence, [\(2\)](#) relation-algebraically specifies the two lattice prerations \sqcap and \sqcup . As a consequence, an ordered set (X, R) constitutes a lattice (X, \sqcup, \sqcap) iff $\mathbf{L} = \text{Inf}(R)\mathbf{L}$ and $\mathbf{L} = \text{Sup}(R)\mathbf{L}$, since the latter equations express that the two relations $\text{Inf}(R)$ and $\text{Sup}(R)$ are total (see [\[11\]](#)). Also complete lattices can easily be characterized by relation-algebraic means. If $R : X \leftrightarrow X$ is the partial order of a lattice (X, \sqcup, \sqcap) , then X is complete iff $\mathbf{L} = \mathbf{L} \text{glb}(R, \mathbf{M})$ or, equivalently, iff $\mathbf{L} = \mathbf{L} \text{lub}(R, \mathbf{M})$, where $\mathbf{M} : X \leftrightarrow 2^X$ is the membership relation.

3 Computing and Visualizing Closure Objects

In the introduction we have mentioned several concepts which we refer to as “closure objects”. In the literature, all of these notions are usually defined on powersets; see e.g., [\[6\]](#). But with the exception of dependency relations the restriction to such a specific class of lattices is not necessary. We therefore prefer to define all closure objects on more general ordered structures. In this section, we develop relation-algebraic specifications for recognizing, computing and transforming closure objects on complete lattices. The specifications will be algorithmic and, hence, can directly be translated into RELVIEW-code. Since RELVIEW only allows to treat relations on finite sets, we assume for the developments finite lattices if this is advantageous.

3.1 Closure Systems

Assume (X, \sqcup, \sqcap) to be a complete lattice. Then $S \subseteq X$ is called a *closure system* (or a Moore family) of X if it is closed under arbitrary least upper bounds, that is, for all $X \subseteq S$ we have $\bigsqcup X \in S$. In the case of a finite carrier set X this second order definition is obviously equivalent to the two requirements that $\top \in S$, where \top denotes the greatest element of the lattice, and that for all $x, y \in S$ also $x \sqcap y \in S$. The next theorem provides the transformation of this first-order specification to relation algebra.

Theorem 3.11. *Assume $R : X \leftrightarrow X$ to be the partial order of a finite lattice (X, \sqcup, \sqcap) and let $S \subseteq X$ be represented by the vector $s : X \leftrightarrow 1$. Then S is a closure system of X iff the following formulae hold:*

$$\text{gel}(R, L) \subseteq s \quad [s^\top, s^\top]^\top \subseteq \text{Inf}(R) s$$

Proof. As $L : X \leftrightarrow 1$ represents the carrier X , the formula $\text{gel}(R, L) \subseteq s$ expresses that S contains the greatest lattice element \top . The following calculation shows that the second formula specifies S to be closed under the binary operation \sqcap .

$$\begin{aligned} & \forall u \in X \times X : u_1 \in S \wedge u_2 \in S \rightarrow u_1 \sqcap u_2 \in S \\ \Leftrightarrow & \forall u \in X \times X : s_{u_1} \wedge s_{u_2} \rightarrow \exists x \in X : \text{Inf}(R)_{u,x} \wedge s_x \\ \Leftrightarrow & \forall u \in X \times X : [s^\top, s^\top]_u^\top \rightarrow (\text{Inf}(R) s)_u \\ \Leftrightarrow & [s^\top, s^\top]^\top \subseteq \text{Inf}(R) s \quad \square \end{aligned}$$

The formulae of Theorem 3.11 can immediately be translated into the programming language of the computer algebra system RELVIEW. Hence, given a partial order $R : X \leftrightarrow X$ of a lattice and a vector $s : X \leftrightarrow 1$, the tool can be used to test whether s represents a closure system. Note that in Theorem 3.11 the type of s^\top is $[1 \leftrightarrow X]$ and, thus, the type of $[s^\top, s^\top]$ is $[1 \leftrightarrow X \times X]$. This is advantageous for the implementation in RELVIEW, which is based on ROBDDs. In general, the transposition of a relation requires that a new ROBDD has to be computed from the old one by exchanging the variables encoding the domain with those encoding the range. But in the case of a relation with domain or range 1 this process can be omitted since the ROBDD of the relation and its transpose coincide [4].

Having specified a single closure system within relation algebra, we turn to specify the set $\mathfrak{S}(X)$ of all closure systems of X as a subset of the powerset via a vector of type $[2^X \leftrightarrow 1]$. In the following theorem $M : X \leftrightarrow 2^X$ is a membership relation, $\pi, \rho : X \times X \leftrightarrow X$ are the projection relations of $X \times X$, the left L has type $[X \leftrightarrow 1]$ and the remaining L is of type $[1 \leftrightarrow X \times X]$.

Theorem 3.12. *Assume again $R : X \leftrightarrow X$ to be the partial order of a finite lattice (X, \sqcup, \sqcap) . Then the following vector represents the set $\mathfrak{S}(X)$ of all closure systems of X as a subset of the powerset 2^X :*

$$\text{cls}(R) := (\text{gel}(R, L)^\top M \cap \overline{L(\pi M \cap \rho M \cap \overline{\text{Inf}(R) M})}^\top) : 2^X \leftrightarrow 1$$

Proof. For all $S \in 2^X$ holds

$$\top \in S \Leftrightarrow \exists x \in X : M_{x,S} \wedge \text{gel}(R, L)_x \Leftrightarrow (M^\top \text{gel}(R, L))_S$$

and also

$$\begin{aligned} & \forall u \in X \times X : u_1 \in S \wedge u_2 \in S \rightarrow u_1 \sqcap u_2 \in S \\ \Leftrightarrow & \forall u \in X \times X : u_1 \in S \wedge u_2 \in S \rightarrow \exists z \in X : u_1 \sqcap u_2 = z \wedge z \in S \\ \Leftrightarrow & \forall u \in X \times X : u_1 \in S \wedge u_2 \in S \rightarrow \exists z \in X : \text{Inf}(R)_{u,z} \wedge z \in S \\ \Leftrightarrow & \forall u \in X \times X : (\pi M)_{u,S} \wedge (\rho M)_{u,S} \rightarrow \exists z \in X : \text{Inf}(R)_{u,z} \wedge M_{z,S} \\ \Leftrightarrow & \forall u \in X \times X : (\pi M)_{u,S} \wedge (\rho M)_{u,S} \rightarrow (\text{Inf}(R) M)_{u,S} \\ \Leftrightarrow & \neg \exists u \in X \times X : (\pi M)_{u,S} \wedge (\rho M)_{u,S} \wedge \overline{\text{Inf}(R) M}_{u,S} \\ \Leftrightarrow & \neg \exists u \in X \times X : (\pi M \cap \rho M \cap \overline{\text{Inf}(R) M})_{S,u}^\top \wedge L_u \\ \Leftrightarrow & \overline{(\pi M \cap \rho M \cap \overline{\text{Inf}(R) M})^\top L}_S. \end{aligned}$$

As a consequence, $M^\top \text{gel}(R, L) \cap \overline{(\pi M \cap \rho M \cap \overline{\text{Inf}(R) M})^\top L}$ represents $\mathfrak{S}(X)$ and the result follows according to three simple rules of transposition, namely $L = L^\top$, $R^\top S^\top = (SR)^\top$, and $\overline{R^\top} = \overline{R}^\top$. \square

Again, the transpositions occurring in the definition of $\text{cls}(R)$ are motivated by the aim to obtain an efficient RELVIEW program. Stating the formula in the given way, the relations affected during program execution are all of domain 1. If, in contrast, we would not have simplified the result by applying the rule $R^\top S^\top = (SR)^\top$ the resulting program would be less efficient and not scale anymore.

A significant fact about the set $\mathfrak{S}(X)$ is that it is itself a closure system of the powerset lattice $(2^V, \cup, \cap)$. Hence, it forms a complete lattice with intersection as greatest lower bound operation and set inclusion as partial order. A relation-algebraic specification of the partial order of $\mathfrak{S}(X)$ is rather simple. Using the technique described in Section 2.3, by

$$\text{ClSys}(R) := M \text{inj}(\text{cls}(R))^\top : X \leftrightarrow \mathfrak{S}(X) \quad (3)$$

the set $\mathfrak{S}(X)$ is enumerated by column and we immediately obtain from (3) that the set inclusion on $\mathfrak{S}(X)$ can be specified as

$$\text{ClLat}(R) := \overline{\text{ClSys}(R)^\top \text{ClSys}(R)} : \mathfrak{S}(X) \leftrightarrow \mathfrak{S}(X). \quad (4)$$

The number of different closure systems of a finite lattice grows very rapidly, see [6] for the numbers in the case of powersets 2^X up to $|X| = 6$. Therefore, a visualization of the according lattice is useful for rather small examples, only.

Example 3.11. Figure 11 contains a picture of the Hasse-diagram of the partial order of a lattice X^* with 6 elements x_1, \dots, x_6 . The picture demonstrates the support in RELVIEW to depict relations as directed graphs. Each vertex with label n represents the lattice element x_n , $1 \leq n \leq 6$.

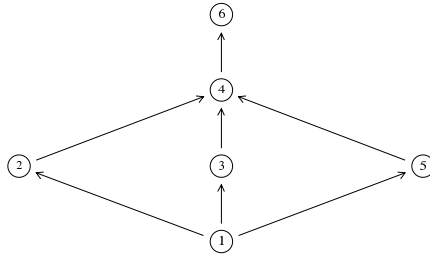


Fig. 1. Hasse-diagram of the partial order of a lattice with 6 elements

Stating the relation-algebraic specifications (3) and (4) above as RELVIEW-programs, we computed that 24 of the 64 subsets of X^* are closure systems. The result of this computation is shown in Figure 2. There the 24 subsets are enumerated by column in a 6×24 Boolean matrix. In the picture a filled square denotes a 1-entry and an empty square a 0-entry. If we denote the closure system represented by column i with S_i , $1 \leq i \leq 24$, then, e.g., the first column represents the closure system $S_1 = \{x_6\}$ consisting of the greatest lattice element only, the second column represents the closure system $S_2 = \{x_5, x_6\}$, the third column represents the closure system $S_3 = \{x_4, x_6\}$ and the last column represents the closure system $S_{24} = X^*$ consisting of all lattice elements.

Finally, Figure 3 shows the Hasse-diagram of the lattice $\mathfrak{G}(X^*)$, where the vertex with label i corresponds to the closure system S_i , $1 \leq i \leq 24$, and an arrow denotes set inclusion. From Figure 2 it follows that the n -th layer of the graph exactly contains the vertices corresponding to the closure systems with cardinality n , $1 \leq n \leq 6$. □

3.2 Closure Operations

For a given partial order (X, R) a *closure operation* is a function $C : X \rightarrow X$ which is extensive, monotone, and idempotent. Note that it is not necessary to restrict such operations to sets. In the next theorem we provide a relation-algebraic characterization of closure operations of (X, R) as specific relations of type $[X \leftrightarrow X]$. Again the given formulae directly lead to a RELVIEW-program for recognizing closure operations.

Theorem 3.21. *Given a partial order $R : X \leftrightarrow X$, a relation $C : X \leftrightarrow X$ is a closure operation of the ordered set (X, R) iff the following formulae hold:*

$$C\bar{1} = \bar{C} \quad C \subseteq R \quad R \subseteq CRC^T \quad CC \subseteq C$$

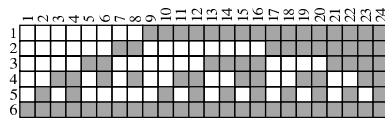


Fig. 2. Closure systems of the partial order of Figure 1

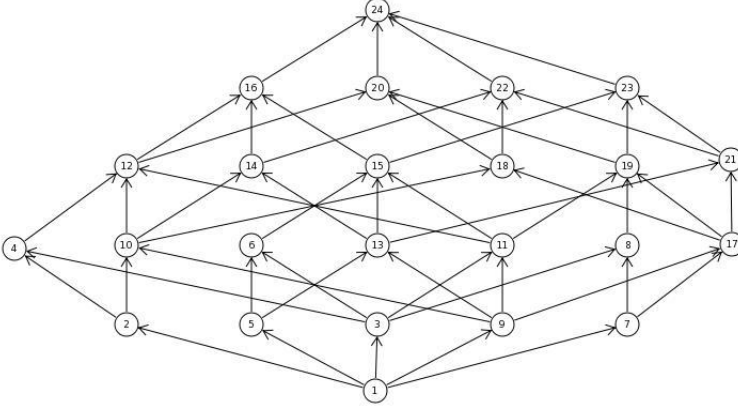


Fig. 3. Hasse-diagram of the lattice $\mathfrak{S}(X^*)$

Proof. The first equation characterizes C as a function; cf. [11]. To enhance readability, we apply the common notation of function application for C in the following. First, we show that $C \subseteq R$ states that C is extensive.

$$\begin{aligned} \forall x \in X : R_{x,C(x)} &\Leftrightarrow \forall x, y \in X : C(x) = y \rightarrow R_{x,y} \\ &\Leftrightarrow \forall x, y \in X : C_{x,y} \rightarrow R_{x,y} \\ &\Leftrightarrow C \subseteq R \end{aligned}$$

The next calculation verifies that $R \subseteq CRC^\top$ specifies monotonicity.

$$\begin{aligned} &\forall x, y \in X : R_{x,y} \rightarrow R_{C(x),C(y)} \\ \Leftrightarrow &\forall x, y \in X : R_{x,y} \rightarrow \exists a, b \in X : C(x) = a \wedge C(y) = b \wedge R_{a,b} \\ \Leftrightarrow &\forall x, y \in X : R_{x,y} \rightarrow \exists a \in X : C_{x,a} \wedge \exists b \in X : R_{a,b} \wedge C_{b,y}^\top \\ \Leftrightarrow &\forall x, y \in X : R_{x,y} \rightarrow (CRC^\top)_{x,y} \\ \Leftrightarrow &R \subseteq CRC^\top \end{aligned}$$

Finally, idempotency and transitivity of C are equivalent due to

$$\begin{aligned} \forall x \in X : C(C(x)) = x &\Leftrightarrow \forall x, y, a \in X : C(x) = a \wedge C(a) = y \rightarrow C(x) = y \\ &\Leftrightarrow \forall x, y, a \in X : C_{x,a} \wedge C_{a,y} \rightarrow C_{x,y} \\ &\Leftrightarrow \forall x, y \in X : (\exists a \in X : C_{x,a} \wedge C_{a,y}) \rightarrow C_{x,y} \\ &\Leftrightarrow \forall x, y \in X : (CC)_{x,y} \rightarrow C_{x,y} \\ &\Leftrightarrow CC \subseteq C. \quad \square \end{aligned}$$

On complete lattices (X, \sqcup, \sqcap) there is a well-known one-to-one correspondence between the set $\mathfrak{D}(X)$ of all closure operations and the set $\mathfrak{S}(X)$ of all closure systems. The closure system corresponding to the closure operation $C \in \mathfrak{D}(X)$ is the set of all fixed points of C . Conversely, the closure operation corresponding to the closure system $S \in \mathfrak{S}(X)$ is such that $x \in X$ is mapped to $\bigcap \{z \in S \mid R_{x,z}\}$, where $R : X \leftrightarrow X$ is the partial order of the lattice. The following theorem states how these correspondences can be formulated as a pair of relation-algebraic

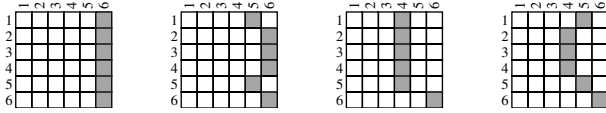


Fig. 4. Closure operations $C_1 \dots C_4$ of $\mathfrak{D}(X^*)$

specifications. To this end, we identify the subsets of X with their representation as vectors $[X \leftrightarrow 1]$. This enables us to consider $\mathfrak{S}(X)$ to as a subset of the powerset $2^{[X \leftrightarrow X]}$, and $\mathfrak{D}(X)$ as a subset of $[X \leftrightarrow X]$.

Theorem 3.22. *Let $R : X \leftrightarrow X$ be the partial order of a lattice (X, \sqcup, \sqcap) . Then, for all $C \in \mathfrak{D}(X)$ the vector*

$$\text{CloToCls}(C) := (C \cap \mathbb{I})\mathbb{L} : X \leftrightarrow 1$$

represents the set of all fixed points of C and for all $s \in \mathfrak{S}(X)$ the relation

$$\text{ClsToClo}(s) := \text{glb}(R, s\mathbb{L} \cap R^\top)^\top : X \leftrightarrow X$$

fulfills for all $x, y \in X$ that $\text{ClsToClo}(s)_{x,y}$ iff $y = \prod\{z \in V \mid s_z \wedge R_{x,z}\}$.

Proof. Applying the function notation for C , we get for all $x \in X$ that

$$C(x) = x \Leftrightarrow \exists y \in X : C_{x,y} \wedge x = y \wedge \mathbb{L}_y \Leftrightarrow ((C \cap \mathbb{I})\mathbb{L})_x$$

and, thus, the definition of $\text{CloToCls}(C)$ ensures the first claim. To prove the second claim, we calculate for given $x, y \in X$

$$\begin{aligned} y = \prod\{z \in V \mid s_z \wedge R_{x,z}\} &\Leftrightarrow y = \prod\{z \in V \mid (s\mathbb{L})_{z,x} \wedge R_{z,x}^\top\} \\ &\Leftrightarrow \text{glb}(R, s\mathbb{L} \cap R^\top)_{y,x} && \text{(cf. Section 2.4)} \\ &\Leftrightarrow \text{glb}(R, s\mathbb{L} \cap R^\top)_{x,y}^\top \end{aligned}$$

and the definition of $\text{ClsToClo}(C)$ yields the desired result. □

The functions $\text{CloToCls} : \mathfrak{D}(X) \rightarrow \mathfrak{S}(X)$ and $\text{ClsToClo} : \mathfrak{S}(X) \rightarrow \mathfrak{D}(X)$ of Theorem 3.22 are order-reversing with respect to set inclusion for closure systems and the pointwise ordering of functions. The latter order on functions can be specified in relation algebra as $C_1 \leq C_2$ iff $C_1 \subseteq C_2 R^\top$.

Example 3.21. As the functions of Theorem 3.22 are order-reversing a transposition of the Hasse-diagram in Figure 3 yields the Hasse-diagram of the lattice $\mathfrak{D}(X^*)$. Accordingly, in the resulting graph the vertex with label n represents the closure operation C_n corresponding to the closure system S_i for $1 \leq i \leq 24$ and Figure 4 depicts the relations $C_1 \dots C_4$. The function C_1 maps all elements to the greatest lattice element. It is the greatest closure operation with respect to the pointwise ordering. The least closure operation is the identity relation \mathbb{I} and corresponds to the greatest closure system S_{24} . Figure 5 shows the partial

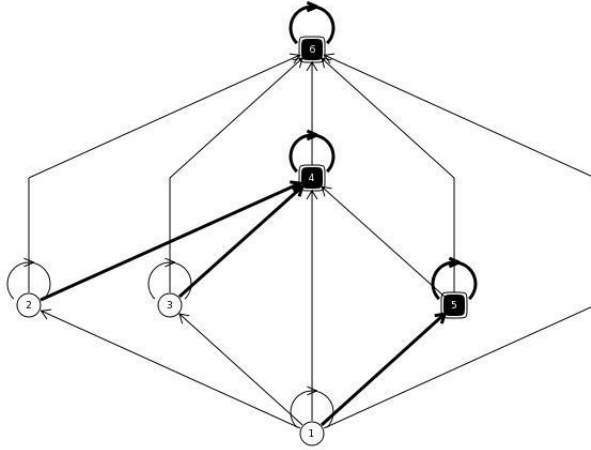


Fig. 5. Lattice X^* , emphasizing closure system S_4 and closure operation C_4

order relation of the lattice X^* as directed graph, where the vertices of the closure system S_4 are emphasized as black squares and the arcs corresponding to the pairs of the closure operation C_4 are drawn boldface.

Two of the three properties of closure operations can immediately be verified by examining the picture. The operation C_4 is extensive, since each arc of C_4 is an arc of the graph. It is idempotent, since each C_4 -path leads into a loop over at most one non-loop edge. Monotonicity of C_4 can be recognized by pointwise comparisons. The picture also clearly visualizes that each element of the lattice is either a fixed point of C_4 , i.e., contained in the corresponding closure system S_4 , or is mapped to the least element of S_4 above it. \square

The *topological closure operations* of a lattice (X, \sqcup, \sqcap) distribute over the \sqcup -operation and form an important subclass of $\mathfrak{D}(X)$. If the lattice X is finite, the corresponding closure systems are precisely the sublattices of X which contain the greatest element of X . Assuming $R : X \leftrightarrow X$ to be the partial order of X , a simple calculation shows that $C \in \mathfrak{D}(X)$ is topological iff

$$(C \parallel C) \text{Sup}(R) = \text{Sup}(R) C. \quad (5)$$

We have transformed (5) into RELVIEW-code and computed for the above example lattice X^* that exactly 20 out of the 24 closure operations are topological. The four exceptions are C_{14}, C_{18}, C_{21} and C_{22} .

3.3 Full Implicational Systems and Join-Congruences

The origin of full implicational systems is relational database theory, where they are called families of functional dependencies (see e.g., [7]). In [6] full implicational systems are defined on powersets by a variant of the well known Armstrong axioms which require for all sets A, B, C, D that 1) if $A \rightarrow B$ and $B \rightarrow C$

then $A \rightarrow C$, 2) if $A \supseteq B$ then $A \rightarrow B$ and 3) if $A \rightarrow B$ and $C \rightarrow D$ then $A \cup C \rightarrow B \cup D$. We generalize this description to finite (complete) lattices (X, \sqcup, \sqcap) with partial order $R : X \leftrightarrow X$. In this sense, a *full implicational system* on X is a relation $F : X \leftrightarrow X$ that is 1) transitive, 2) contains R^\top and 3) for all $x, y, x', y' \in X$ it holds that $F_{x,x'}$ and $F_{y,y'}$ imply $F_{x \sqcup y, x' \sqcup y'}$. As a side remark we note that axiom 3) could be generalized to arbitrary least upper bounds, i.e., arbitrary complete lattices. The resulting relation-algebraic formulation would be that 3') for all sublattices $D \subseteq F$ we have $vw^\top \subseteq F$, where $v := \text{lub}(R, DL)$ specifies the least upper bound of all first components of pairs of D and $w := \text{lub}(R, D^\top L)$ does the same for the second components. But as we restrict ourselves to finite relations, the following theorem considers the first version of the axiom only.

Theorem 3.31. *Given $R : X \leftrightarrow X$ as partial order of a finite lattice (X, \sqcup, \sqcap) , $F : X \leftrightarrow X$ is a full implicational system of X iff the following formulae hold:*

$$FF \subseteq F \quad R^\top \subseteq F \quad F \parallel F \subseteq \text{Sup}(R) F \text{Sup}(R)^\top$$

Proof. We only consider the last formula and get the desired property by

$$\begin{aligned} & \forall u, v \in X \times X : F_{u_1, v_1} \wedge F_{u_2, v_2} \rightarrow F_{u_1 \sqcup u_2, v_1 \sqcup v_2} \\ \Leftrightarrow & \forall u, v \in X \times X : (F \parallel F)_{u, v} \rightarrow \exists x, y \in X : x = u_1 \sqcup u_2 \wedge y = v_1 \sqcup v_2 \wedge F_{x, y} \\ \Leftrightarrow & \forall u, v \in X \times X : (F \parallel F)_{u, v} \rightarrow \exists x, y \in X : \text{Sup}(R)_{u, x} \wedge F_{x, y} \wedge \text{Sup}(R)_{v, y} \\ \Leftrightarrow & \forall u, v \in X \times X : (F \parallel F)_{u, v} \rightarrow (\text{Sup}(R) F \text{Sup}(R)^\top)_{u, v} \\ \Leftrightarrow & F \parallel F \subseteq \text{Sup}(R) F \text{Sup}(R)^\top. \quad \square \end{aligned}$$

If full implicational systems are ordered by inclusion, then the complete lattice induced by $(\mathfrak{D}(X), \leq)$ is isomorphic to the complete lattice induced by $(\mathfrak{F}(X), \subseteq)$, where $\mathfrak{F}(X)$ denotes the set of all full implicational systems of (X, \sqcup, \sqcap) . One direction of this isomorphism is given by mapping $C \in \mathfrak{D}(X)$ to the full implicational system $F \in \mathfrak{F}(X)$ that consists of all pairs $\langle x, y \rangle \in X \times X$ with $R_{y, C(x)}$. The converse direction is obtained by mapping $F \in \mathfrak{F}(X)$ to the closure operation $C \in \mathfrak{D}(X)$ such that $C(x) = \bigsqcup \{z \in X \mid D_{x, z}\}$ for all $x \in X$. The following theorem yields these correspondences formulated as a pair of relation-algebraic specifications.

Theorem 3.32. *Let $R : X \leftrightarrow X$ be the partial order of a lattice (X, \sqcup, \sqcap) . Then, for all $C \in \mathfrak{D}(X)$ the relation*

$$\text{CloToFis}(C) := CR^\top : X \leftrightarrow X$$

fulfills for all $x, y \in X$ that $\text{CloToFis}(C)_{x, y}$ iff $R_{y, C(x)}$, and, conversely, for all $F \in \mathfrak{F}(X)$ the relation

$$\text{FisToClo}(F) := \text{lub}(R, F^\top)^\top : X \leftrightarrow X$$

fulfills for all $x, y \in X$ that $\text{FisToClo}(F)_{x, y}$ iff $y = \bigsqcup \{z \in X \mid F_{x, z}\}$.

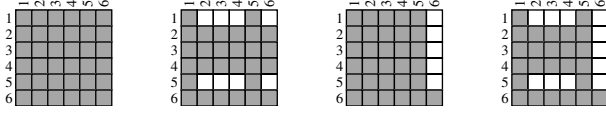


Fig. 6. Full implicational systems F_1 to F_4

Proof. The first claim follows from the definition of $\text{CloToFis}(C)$ and

$$R_{y,C(x)} \Leftrightarrow \exists z \in X : C_{x,z} \wedge R_{z,y}^\top \Leftrightarrow (CR^\top)_{x,y}$$

and the proof of the second claim is analogue to the one of Theorem 3.22. \square

There is a very close relation between full implicational systems and join-congruence relations, which are generalizations of lattice congruences. Given a lattice (X, \sqcup, \sqcap) , a relation $J : X \leftrightarrow X$ is a *join-congruence* of X iff it is an equivalence relation and, in addition for all $x, y, z \in X$ from $J_{x,y}$ it follows that $J_{x \sqcup z, y \sqcup z}$. How to specify equivalence relations with relation-algebraic means is well-known; see e.g., [11]. From the proof of Theorem 3.31 we obtain as special case that the remaining requirement on join-congruences holds for J iff $J \parallel \text{I} \subseteq \text{Sup}(R) J \text{Sup}(R)^\top$. This leads to the following result.

Theorem 3.33. *Let $R : X \leftrightarrow X$ be the partial order of a lattice (X, \sqcup, \sqcap) . Then $J : X \leftrightarrow X$ is a join-congruence of X iff the following formulae hold:*

$$\text{I} \subseteq J \quad J = J^\top \quad JJ \subseteq J \quad J \parallel \text{I} \subseteq \text{Sup}(R) J \text{Sup}(R)^\top \quad \square$$

In the case of a finite lattice (X, \sqcup, \sqcap) there is a one-to-one correspondence between the set $\mathfrak{D}(X)$ of all closure operations of X and the set $\mathfrak{J}(X)$ of all join-congruences of X which again establishes a lattice isomorphism wrt. the lattices induced by the ordered sets $(\mathfrak{D}(X), \leq)$ and $(\mathfrak{J}(X), \subseteq)$. The join-congruence J associated with the closure operation $C \in \mathfrak{D}(X)$ is the kernel of the function C , i.e., we have for all $x, y \in X$ that $J_{x,y}$ iff $C(x) = C(y)$. Relation-algebraically this means that $J = \text{CloToJc}(C)$, where

$$\text{CloToJc}(C) := CC^\top : X \leftrightarrow X. \quad (6)$$

In the reverse direction, the closure operation C is obtained from the join-congruence $J \in \mathfrak{J}(X)$ as in the case of full implicational systems, i.e., by mapping each element $x \in X$ to $\bigsqcup\{z \in X \mid J_{x,z}\}$. Using Theorem 3.32 we get $C = \text{JcToClo}(J)$ as

$$\text{JcToClo}(J) := \text{lub}(R, J)^\top : X \leftrightarrow X \quad (7)$$

where $R : X \leftrightarrow X$ is the partial order of the finite lattice (X, \sqcup, \sqcap) .

Example 3.31. In Figure 6 the full implicational systems F_1 to F_4 of our running example are shown as RELVIEW-pictures, where F_i is the value of $\text{CloToFis}(C_i)$ with the closure operations C_i from Figure 4, $1 \leq i \leq 4$.

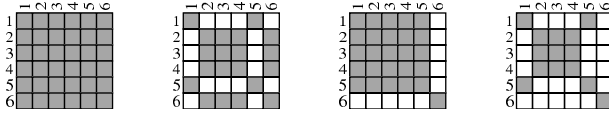


Fig. 7. Join congruences J_1 to J_4

For our running example we already know that exactly 24 equivalence relations $J_i = \text{CloToJc}(C_i)$, $1 \leq i \leq 24$, on X^* are join-congruences. Figure 7 shows the relations J_1 to J_4 . Each column (or row) directly corresponds to a congruence class of the respective relation. In addition we used RELVIEW to test which of the 24 join-congruences are also meet-congruences. Analogously, a meet-congruence M satisfies for all $x, y, z \in X$ that $M_{x,y}$ implies $M_{x \sqcap z, y \sqcap z}$. We obtained four positive answers: J_1, J_3, J_{22} (with the classes $\{x_1\}, \{x_2\}, \{x_3\}, \{x_4, x_6\}, \{x_5\}$) and $\perp = J_{24}$. The relation J_2 , for example, is not a meet-congruence, since $x = z = x_2$ and $y = x_3$ is one of the 12 triples such that $\langle x, y \rangle$ is in J_2 but $\langle x \sqcap z, y \sqcap z \rangle$ is not in J_2 . \square

3.4 Dependency Relations

In [1] Aumann introduced certain relations to formalize the essential properties of a “contact” between objects and sets of objects. His motivation was to obtain an access to topology which is more suggestive for beginners than the ones provided by “traditional” axiom systems. If we formulate his original definition in our notation, then a relation $D : X \leftrightarrow 2^X$ is a *contact* if 1) for all $x \in X$ and $Y, Z \in 2^X$ we have $D_{x, \{x\}}$, that 2) from $D_{x,Y}$ and $Y \subseteq Z$ it follows $D_{x,Z}$, and that 3) from $D_{x,Y}$ it follows $D_{x,Z}$ if $D_{y,Z}$ holds for all $y \in Y$. Obviously, demands 1) and 2) are equivalent to the fact that $x \in Y$ implies $D_{x,Y}$ for all $x \in X$ and $Y \in 2^X$. Hence, Aumann’s contacts are exactly the *dependency relations* in the sense of [6]. In the following theorem, we present relation-algebraic versions of the two axioms given in [6].

Theorem 3.41. *Let $M : X \leftrightarrow 2^X$ be a membership-relation. Then a relation $D : X \leftrightarrow 2^X$ is a dependency relation iff the following formulae hold:*

$$M \subseteq D \quad \overline{DM^T \overline{D}} \subseteq D$$

Proof. We only show how the second inclusion can be obtained from a first-order formalization of the second axiom of [6]:

$$\begin{aligned} & \forall x \in X, Y, Z \in 2^X : D_{x,Y} \wedge (\forall y \in Y : D_{y,Z}) \rightarrow D_{x,Z} \\ \Leftrightarrow & \forall x \in X, Y, Z \in 2^X : D_{x,Y} \wedge \neg(\exists y \in X : y \in Y \wedge \overline{D}_{y,Z}) \rightarrow D_{x,Z} \\ \Leftrightarrow & \forall x \in X, Y, Z \in 2^X : D_{x,Y} \wedge \overline{M^T \overline{D}}_{Y,Z} \rightarrow D_{x,Z} \\ \Leftrightarrow & \forall x \in X, Z \in 2^X : (\exists Y \in 2^X : D_{x,Y} \wedge \overline{M^T \overline{D}}_{Y,Z}) \rightarrow D_{x,Z} \\ \Leftrightarrow & \forall x \in X, Z \in 2^X : (D \overline{M^T \overline{D}})_{x,Z} \rightarrow D_{x,Z} \\ \Leftrightarrow & D \overline{M^T \overline{D}} \subseteq D \end{aligned}$$

\square

In [1] a one-to-one correspondence between the set $\mathfrak{D}(2^X)$ of all closure operations of $(2^X, \subseteq)$ and the set $\mathfrak{D}(X)$ of all contacts of type $[X \leftrightarrow 2^X]$ is established, which is also mentioned in [6] for dependency relations. The relation $D \in \mathfrak{D}(X)$ corresponding to $C \in \mathfrak{D}(2^X)$ is for all $x \in X$ and $Y \in 2^X$ given by $D_{x,Y}$ iff $x \in C(Y)$. Conversely, the closure operation associated with $D \in \mathfrak{D}(X)$ maps $Y \in 2^X$ to $\{x \in X \mid D_{x,Y}\}$. The next theorem contains corresponding specifications in relation algebra.

Theorem 3.42. *Assume $M : X \leftrightarrow 2^X$ to be a membership-relation. Then, for all $C \in \mathfrak{D}(2^X)$ the relation*

$$\text{CloToDep}(C) := MC^T : X \leftrightarrow 2^X$$

fulfills for all $x \in X$ and $Y \in 2^X$ that $\text{CloToDep}(C)_{x,Y}$ iff $x \in C(Y)$, and, conversely, for all $D \in \mathfrak{D}(X)$ the relation

$$\text{DepToClo}(D) := \text{syq}(D, M) : 2^X \leftrightarrow 2^X$$

fulfills for all $Y \in X$ that $\text{DepToClo}(D)(Y) = \{x \in X \mid D_{x,Y}\}$.

Proof. For the first claim, we calculate for all $x \in X$ and $Y \in 2^X$ that

$$x \in C(Y) \Leftrightarrow \exists Z \in 2^X : M_{x,Z} \wedge C_{Z,Y}^T \Leftrightarrow (MC^T)_{x,Y}$$

and use the definition of $\text{CloToDep}(C)_{x,Y}$. Since for all $Y, Z \in 2^X$ we have

$$\begin{aligned} \{x \in X \mid D_{x,Y}\} = Z &\Leftrightarrow \forall x \in X : D_{x,Y} \leftrightarrow x \in Z \\ &\Leftrightarrow \forall x \in X : D_{x,Y} \leftrightarrow M_{x,Z} \\ &\Leftrightarrow \text{syq}(D, M)_{Y,Z} \end{aligned} \quad (\text{cf. Section 2})$$

the definition of $\text{DepToClo}(D)_{Y,Z}$ in combination with the common notation for function application yields the second claim. \square

In [1] Aumann mentions that his relations may also be used to investigate the notion of a contact in sociology or political science. For this, it is frequently necessary to replace M by a relation $M : X \leftrightarrow G$ with the interpretation “individual x is a member of a group g of individuals” of $M_{x,g}$. If $\text{syq}(M, M) = 1$, then (G, R) is an ordered set, where $R := \overline{M^T M}$. In this general setting the one-to-one correspondence between closure operations and contacts is lost. It can only be shown that there is an order embedding from the set of closure operations to the set of these generalized contacts.

4 Conclusion

Closure systems and closure operations play an important role in both mathematics and computer science. Moreover, the literature contains many examples for concepts employed in practice which could be proven to be isomorphic to special closure systems. In this work we have presented relation-algebraic formulations of the connections between closure systems on the one hand and

closure operations, full implication systems, join-congruences, and dependency relations on the other hand. The resulting algebraic representations are very compact and we have given proofs of the correctness of each formulation. In addition we have demonstrated that the formulas can directly be used to compute the transformation between concepts, e.g., transform a given finite dependency relation to the corresponding closure operation. Each of the definitions can also be used to efficiently test given relations for conformance, e.g., to compute whether a given relation is a dependency relation or not. We have used the computer algebra system RELVIEW to compute both transformations and tests. As shown by examples, the system can also be used to visualize the results either as graphs or as Boolean matrices, whatever is more appropriate to the case.

A final word about scalability. We have taken care that the presented tests and transformations fit well into the setting of relations implemented using ROBDDs. As a consequence all of the resulting programs scale well and are also applicable to big relations with ten thousands of elements. The only exception is the enumeration of all possible closure systems on a given set provided with Theorem 3.12. As shown in [6] the number of such systems grows too rapidly to be subject to an efficient complete enumeration.

References

1. Aumann, G.: Contact relations. Bayerische Akademie der Wissenschaften, Mathe-Nat. Klasse Sitzungsberichte, 67–77 (1970) (in German)
2. Behnke, R., Berghammer, R., Meyer, E., Schneider, P.: RELVIEW - A system for calculating with relations and relational programming. In: Astesiano, E. (ed.) ETAPS 1998 and FASE 1998. LNCS, vol. 1382, pp. 318–321. Springer, Heidelberg (1998)
3. Berghammer, R.: Solving algorithmic problems on orders and lattices by relation algebra and RELVIEW. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2006. LNCS, vol. 4194, pp. 49–63. Springer, Heidelberg (2006)
4. Berghammer, R., Leoniuk, B., Milanese, U.: Implementation of relational algebra using binary decision diagrams. In: de Swart, H. (ed.) RelMiCS 2001. LNCS, vol. 2561, pp. 241–257. Springer, Heidelberg (2002)
5. Berghammer, R., Neumann, F.: RelView – An OBDD-based Computer Algebra system for relations. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2005. LNCS, vol. 3718, pp. 40–51. Springer, Heidelberg (2005)
6. Caspard, N., Monjardet, B.: The lattices of closure systems, closure operators, and implicational systems on a finite set: a survey. *Discr. Appl. Math.* 127, 241–269 (2003)
7. Demetrovics, J., Lipkin, L.O., Muchnik, J.B.: Functional dependencies in relational databases: a lattice point of view. *Discr. Appl. Math.* 40, 155–185 (1992)
8. Doinon, J.P., Falmagne, J.C.: Knowledge spaces. Springer, Heidelberg (1999)
9. Ganter, B., Wille, R.: Formal concept analysis, Mathematical foundations. Springer, Heidelberg (1998)
10. Leoniuk, B.: ROBDD-based implementation of relational algebra with applications. Ph.D. thesis, Univ. Kiel (2001) (in German)
11. Schmidt, G., Ströhlein, T.: Relations and graphs. *Discrete Mathematics for Computer Scientists*. In: EATCS Monographs on Theor. Comp. Sci., Springer, Heidelberg (1993)
12. Tarski, A.: On the calculus of relations. *J. Symb. Logic* 6, 73–89 (1941)

On Integrability of a Planar ODE System Near a Degenerate Stationary Point

Alexander Bruno¹ and Victor Edneral^{2,*}

¹ Keldysh Institute for Applied Mathematics of RAS
Miusskaya Sq. 4, Moscow, 125047, Russia
bruno@keldysh.ru

² Skobeltsyn Institute of Nuclear Physics
of Lomonosov Moscow State University
Leninskie Gory 1, Moscow, 119991, Russia
edneral@theory.sinp.msu.ru

Abstract. We consider an autonomous system of ordinary differential equations, which is solved with respect to derivatives. To study local integrability of the system near a degenerate stationary point, we use an approach based on Power Geometry method and on the computation of the resonant normal form. For a planar 5-parametric example of such system, we found the complete set of necessary and sufficient conditions on parameters of the system for which the system is locally integrable near a degenerate stationary point.

Keywords: ordinary differential equations, local integrability, resonant normal form, power geometry, computer algebra.

1 Introduction

We consider an autonomous system of ordinary differential equations

$$dx_i/dt \stackrel{\text{def}}{=} \dot{x}_i = \varphi_i(X), \quad i = 1, \dots, n, \quad (1)$$

where $X = (x_1, \dots, x_n) \in \mathbb{C}^n$ and $\varphi_i(X)$ are polynomials.

In a neighborhood of the stationary point $X = X^0$, the system (1) is *locally integrable* if it has there sufficient number m of independent first integrals of the form

$$a_j(X)/b_j(X), \quad j = 1, \dots, m,$$

where functions $a_j(X)$ and $b_j(X)$ are analytic in a neighborhood of the point $X = X^0$. Otherwise we call the system (1) *locally nonintegrable* in this neighborhood.

In [1], there was proposed a method of analysis of integrability of a system based on power transformations and computation of normal forms near stationary solutions of transformed systems [2].

* The grant of the Russian Foundation for Basic Research 08-01-00082 and the grant of the President of the Russian Federation for support of scientific schools 195.2008.2.

In this paper we demonstrate how this approach can be applied to the study of local integrability of the planar case (i.e., $n = 2$) of system (II) near the stationary point $X^0 = 0$ of high degeneracy.

In the neighborhood of the stationary point $X = 0$, system (II) can be written in the form

$$\dot{X} = A X + \tilde{\Phi}(X), \tag{2}$$

where $\tilde{\Phi}(X)$ has no linear in X terms.

Let $\lambda_1, \lambda_2, \dots, \lambda_n$ be eigenvalues of the matrix A . If at least one of them $\lambda_i \neq 0$, then the stationary point $X = 0$ is called an *elementary* stationary point. In this case, system (II) has a normal form which is equivalent to a system of lower order [3]. If all eigenvalues vanish, then the stationary point $X = 0$ is called a *nonelementary* stationary point. In this case, there is no normal form for the system (II). But by using power transformations, a nonelementary stationary point $X = 0$ can be blown up to a set of elementary stationary points. After that, it is possible to compute the normal form and verify that the condition A is satisfied [4] at each elementary stationary point.

If $n = 2$ then rationality of the ratio λ_1/λ_2 and the condition A (see the next paragraph) are necessary and sufficient conditions for local integrability of a system near an elementary stationary point. For local integrability of original system (II) near a degenerate (nonelementary) stationary point, it is necessary and sufficient to have local integrability near each of elementary stationary points, which are produced by the blowing up process described above.

So we study the system

$$\begin{aligned} \dot{x}_1 &= x_1 \sum \phi_Q X^Q, \\ \dot{x}_2 &= x_2 \sum \psi_Q X^Q, \end{aligned} \tag{3}$$

where $Q = (q_1, q_2)$, $X^Q = x_1^{q_1} x_2^{q_2}$; ϕ_Q and ψ_Q are constant coefficients, which can be polynomials in parameters of the system.

System (3) has a quasi-homogeneous initial approximation if there exists an integer vector $R = (r_1, r_2) > 0$ and a number s such that the scalar product

$$\langle Q, R \rangle \stackrel{\text{def}}{=} q_1 r_1 + q_2 r_2 \geq s = \text{const}$$

for nonzero $|\phi_Q| + |\psi_Q| \neq 0$, and between vectors Q with $\langle Q, R \rangle = s$ there are vectors of the form $(q_1, -1)$ and $(-1, q_2)$. In this case, system (3) takes the form

$$\begin{aligned} \dot{x}_1 &= x_1[\phi_s(X) + \phi_{s+1}(X) + \phi_{s+2}(X) + \dots], \\ \dot{x}_2 &= x_2[\psi_s(X) + \psi_{s+1}(X) + \psi_{s+2}(X) + \dots], \end{aligned}$$

where $\phi_k(X)$ is the sum of terms $\phi_Q X^Q$ for which $\langle Q, R \rangle = k$. And the same holds for the $\psi_k(X)$. Then the initial approximation of (3) is the quasi-homogeneous system

$$\begin{aligned} \dot{x}_1 &= x_1 \phi_s(X), \\ \dot{x}_2 &= x_2 \psi_s(X). \end{aligned} \tag{4}$$

We study the problem: what are the conditions on parameters under which system (3) is locally integrable. The local integrability of (4) is necessary for

this. For an autonomous planar system $m = 1$ (on m see the introduction); so there are two cases:

1. System (4) is Hamiltonian, i.e., it has the form

$$\dot{x}_1 = \partial H(X)/\partial x_2, \quad \dot{x}_2 = -\partial H(X)/\partial x_1,$$

where $H(X)$ is a quasi-homogeneous polynomial.

2. System (4) is non Hamiltonian, but it has the first integral $F(X)$:

$$\frac{\partial F(X)}{\partial x_1} x_1 \phi_s + \frac{\partial F(X)}{\partial x_2} x_2 \psi_s = 0,$$

where $F(X)$ is a quasi-homogeneous polynomial.

For the first case, with the additional assumption that the polynomial $H(X)$ is expandable into the product of only square free factors, the problem is solved in [5]. Therefore, here we discuss only the second case. More precisely, we study the system with $R = (2, 3)$ and $s = 7$.

At $R = (2, 3)$ and $s = 7$ the quasi-homogeneous system (4) has the form

$$\dot{x} = a y^3 + b x^3 y, \quad \dot{y} = c x^2 y^2 + d x^5, \tag{5}$$

where $a \neq 0$ and $d \neq 0$.

Lemma 11. *If system (5) with $b \neq 0$ and $c \neq 0$ has the first integral*

$$I = \alpha y^4 + \beta x^3 y^2 + \gamma x^6, \quad \beta \neq 0, \tag{6}$$

then

$$(a d - b c)(3 b + 2 c) = 0. \tag{7}$$

Proof. A derivative of integral (6) with respect to system (5) has the form

$$\begin{aligned} & \partial I / \partial x (a y^3 + b x^3 y) + \partial I / \partial y (c x^2 y^2 + d x^5) = \\ & = (3 \beta a + 4 \alpha c) x^2 y^5 + (6 \gamma a + 3 \beta b + 2 \beta c + 4 \alpha d) x^5 y^3 + \\ & + (6 \gamma b + 2 \beta d) x^8 y \equiv 0, \end{aligned}$$

thus, coefficients at three monomials $x^p y^q$ are equal to zero, i.e.

$$\begin{aligned} 3 \beta a + 4 \alpha c &= 0, & 6 \gamma b + 2 \beta d &= 0, \\ 6 \gamma a + 3 \beta b + 2 \beta c + 4 \alpha d &= 0. \end{aligned} \tag{8}$$

From the first two equations (8), we obtain

$$\alpha = -\frac{3 \beta a}{4 c}, \quad \gamma = -\frac{\beta d}{3 b}. \tag{9}$$

Substituting these values in the third equations (8), cancelling the factor β , multiplying $(b c)$, and simplifying we obtain equality (7).

In accordance with Lemma [\(11\)](#), system [\(5\)](#) has the first integral [\(6\)](#) in the two cases:

1. $3b + 2c = 0$, then in accordance with equalities [\(9\)](#) integral [\(6\)](#) has the form

$$I = \left(-\frac{3}{2}ay^4 + 2cx^3y^2 + dx^6\right)\frac{\beta}{2c} \tag{10}$$

and Hamiltonian function $H = -Ic/(3\beta)$;

2. $ad - bc = 0$, if $3b + 2c \neq 0$, then the integral $c_0 I$ is not a Hamiltonian function for any constant c_0 ; if $3b + 2c = 0$, then integral [\(10\)](#) and Hamiltonian are proportional to the square $(c_1y^2 + c_2x^3)^2$, where $c_1, c_2 = \text{const}$.
 Multiplying x and y in system [\(5\)](#) by the constants, we can reduce 2 from 4 parameters a, b, c, d . For example it is possible to take $a = d = 1$.

In [\[5\]](#), systems [\(3\)](#), [\(5\)](#) were studied in the case 1 above. We study them in the case 2.

2 About Normal Form and the Condition A

Let the linear transformation

$$X = BY \tag{11}$$

bring the matrix A to the Jordan form $J = B^{-1}AB$ and [\(2\)](#) to

$$\dot{Y} = JY + \tilde{\Phi}(Y). \tag{12}$$

Let the formal change of coordinates

$$Y = Z + \Xi(Z), \tag{13}$$

where $\Xi = (\xi_1, \dots, \xi_n)$ and $\xi_j(Z)$ are formal power series, transform [\(12\)](#) in the system

$$\dot{Z} = JZ + \Psi(Z). \tag{14}$$

We write it in the form

$$\dot{z}_j = z_j g_i(Z) = z_j \sum g_{jQ} Z^Q \text{ over } Q \in \mathbb{N}_j, j = 1, \dots, n, \tag{15}$$

where $Q = (q_1, \dots, q_n)$, $Z^Q = z_1^{q_1} \dots z_n^{q_n}$,

$$\mathbb{N}_j = \{Q : Q \in \mathbb{Z}^n, Q + E_j \geq 0\}, j = 1, \dots, n,$$

E_j means the unit vector. Denote

$$\mathbb{N} = \mathbb{N}_1 \cup \dots \cup \mathbb{N}_n. \tag{16}$$

The diagonal $\Lambda = (\lambda_1, \dots, \lambda_n)$ of J consists of eigenvalues of the matrix A .

System [\(14\)](#), [\(15\)](#) is called the *resonant normal form* if:

- a) J is the Jordan matrix,
- b) in writing [\(15\)](#), there are only the *resonant terms*, for which the scalar product

$$\langle Q, \Lambda \rangle \stackrel{\text{def}}{=} q_1\lambda_1 + \dots + q_n\lambda_n = 0. \tag{17}$$

Theorem 21 (Bruno [4]). *There exists a formal change (13) reducing (12) to its normal form (14), (15).*

In [4] there are conditions on the normal form (15), which guarantee the convergence of the normalizing transformation (13).

Condition A. *In the normal form (15)*

$$g_j = \lambda_j \alpha(Z) + \bar{\lambda}_j \beta(Z), \quad j = 1, \dots, n,$$

where $\alpha(Z)$ and $\beta(Z)$ are some power series.

Let

$$\omega_k = \min |\langle Q, \Lambda \rangle| \text{ over } Q \in \mathbb{N}, \langle Q, \Lambda \rangle \neq 0, \sum_{j=1}^n q_j < 2^k, \quad k = 1, 2, \dots$$

Condition ω (on small divisors). *The series*

$$\sum_{k=1}^{\infty} 2^{-k} \log \omega_k > -\infty,$$

i.e. it converges.

It is fulfilled for almost all vectors Λ .

Theorem 22 (Bruno [4]). *If vector Λ satisfies Condition ω and the normal form (2.6) satisfies Condition A then the normalizing transformation (13) converges.*

The algorithm of a calculation of the normal form, the normalizing transformation, and the corresponding computer program are briefly described in [7].

3 The Simplest Nontrivial Example

We consider the system

$$\begin{aligned} dx/dt &= -y^3 - b x^3 y + a_0 x^5 + a_1 x^2 y^2, \\ dy/dt &= (1/b) x^2 y^2 + x^5 + b_0 x^4 y + b_1 x y^3, \end{aligned} \tag{18}$$

with arbitrary complex parameters a_i, b_i and $b \neq 0$.

Systems with a nilpotent matrix of the linear part are thoroughly studied by Lyapunov et. al. In our example, there is no linear part, and the first approximation is not homogeneous but quasi homogeneous. This is the simplest case of a planar system without linear part with Newton's open polygon consisting of a single edge. In our case, the system corresponds to the quasi homogeneous first approximation with $R = (2, 3)$, $s = 7$. In general case, such problems have not been studied, and the authors do not know of any applications of the system (18).

After the power transformation

$$x = u v^2, \quad y = u v^3 \tag{19}$$

and time rescaling

$$dt = u^2 v^7 d\tau,$$

we obtain system (18) in the form

$$\begin{aligned} du/d\tau &= -3u - [3b + (2/b)]u^2 - 2u^3 + (3a_1 - 2b_1)u^2v + (3a_0 - 2b_0)u^3v, \\ dv/d\tau &= v + [b + (1/b)]uv + u^2v + (b_1 - a_1)uv^2 + (b_0 - a_0)u^2v^2. \end{aligned} \tag{20}$$

Under the power transformation (19), the point $x = y = 0$ blows up into two straight lines $u = 0$ and $v = 0$. Along the line $u = 0$, system (20) has a single stationary point $u = v = 0$. Along the second line $v = 0$ this system has three elementary stationary points

$$u = 0, \quad u = -\frac{1}{b}, \quad u = -\frac{3b}{2}. \tag{21}$$

Lemma 31. *Near the point $u = v = 0$, the system (20) is locally integrable.*

Proof. In accordance with Chapter 2 of the book [3], the support of the system (20) consists of the five points $Q = (q_1, q_2)$

$$(0, 0), \quad (1, 0), \quad (2, 0), \quad (1, 1), \quad (2, 1). \tag{22}$$

At the point $u = v = 0$, the eigenvalues of system (20) are $\Lambda = (\lambda_1, \lambda_2) = (-3, 1)$. Only for the first point from (22) $Q = 0$, the scalar product $\langle Q, \Lambda \rangle$ is zero, for the remaining four points (22) it is negative, so these four points lie on the same side of the straight line $\langle Q, \Lambda \rangle = 0$. In accordance with the remark at the end of Subsection 2.1 of Chapter 2 of the book [3], in such case the normal form consists only of the terms of a right-hand side of system (20) such that their support Q lies on the straight line $\langle Q, \Lambda \rangle = 0$. But only linear terms of system (20) satisfy this condition. Therefore, at the point $u = v = 0$ the normal form of the system is linear

$$dz_1/d\tau = -3z_1, \quad dz_2/d\tau = z_2.$$

It is obvious that this normal form satisfies the condition A. So the normalizing transformation converges, and at the point $u = v = 0$ the system (20) has the analytic first integral

$$z_1 z_2^3 = \text{const.}$$

The proof of local integrability at the point $u = \infty, v = 0$ is similar.

Thus, we must find conditions of local integrability at two other stationary points (21). We will have the conditions of local integrability of system (18) near the point $X = 0$.

Let us consider the stationary point $u = -1/b, v = 0$. Below we restrict ourselves to the case $b^2 \neq 2/3$ when a linear part of the system (20), after the

shift $u = \tilde{u} - 1/b$, has non-vanishing eigenvalues. At $b^2 = 2/3$ the shifted system in new variables \tilde{u} and v has a Jordan cell with both zero eigenvalues as the linear part. This case can be studied by using one more power transformation.

To simplify eigenvalues, we change the time at this point once more with the factor $d\tau = (2 - 3b^2)/b^2 d\tau_1$. After that we obtain the vector of eigenvalues of system (20) at this point as $(\lambda_1, \lambda_2) = (-1, 0)$. So the normal form of the system will become

$$\begin{aligned} dz_1/d\tau_1 &= -z_1 + z_1 g_1(z_2), \\ dz_2/d\tau_1 &= z_2 g_2(z_2), \end{aligned} \tag{23}$$

where $g_{1,2}(x)$ are formal power series in x . Coefficients of these series are rational functions of the parameters of the system a_0, a_1, b_0, b_1 and b . It can be proved that denominator of each of these rational functions is proportional to some integer degree $k(n)$ of the polynomial $(2 - 3b^2)$. Their numerators are polynomials in parameters of the system

$$g_{1,2}(x) = \sum_{n=1}^{\infty} \frac{p_{1,2;n}(b, a_0, a_1, b_0, b_1)}{(2 - 3b^2)^{k(n)}} x^n.$$

The condition A of integrability for equation (23) is $g_2(x) \equiv 0$. It is equivalent to the infinite polynomial system of equations

$$p_{2,n}(b, a_0, a_1, b_0, b_1) = 0, \quad n = 1, 2, \dots \tag{24}$$

According to the Hilbert's theorem on bases in polynomial ideals [6], this system has a finite basis.

We computed the first three polynomials $p_{2,1}, p_{2,2}, p_{2,3}$ by our program [7]. There are 2 solutions of a corresponding subset of equations (24) at $b \neq 0$

$$a_0 = 0, \quad a_1 = -b_0 b, \quad b_1 = 0, \quad b^2 \neq 2/3 \tag{25}$$

and

$$a_0 = a_1 b, \quad b_0 = b_1 b, \quad b^2 \neq 2/3. \tag{26}$$

Addition of the fourth equation $p_{2,4} = 0$ to the subset of equations does not change these solutions.

The calculation of polynomials $p_{2,n}(b, a_0, a_1, b_0, b_1)$ in generic case is technically a very difficult problem. But we can verify some of these equations from the set (24) on solutions (25) and (26) for several fixed values of the parameter b . We verified solutions of subset of equations

$$p_{2,n}(b, a_0 = a_1 b, a_1, b_0 = b_1 b, b_1) = 0, \quad n = 1, 2, \dots, 28.$$

at $b = 1$ and $b = 2$. All equations are satisfied, so we can assume that (25) and (26) satisfy the condition A at the stationary point $u = -1/b, v = 0$.

Let us consider the stationary point $u = -3b/2, v = 0$. We rescale time at this point with the factor $d\tau = (2 - 3b^2) d\tau_2$. After that we get the vector of

eigenvalues of the system (20) at this point as $(-1/4, 3/2)$. So the normal form has a resonance of the seventh order

$$\begin{aligned} dz_1/d\tau_2 &= -(1/4) z_1 + z_1 r_1(z_1^6 z_2), \\ dz_2/d\tau_2 &= (3/2) z_2 + z_2 r_2(z_1^6 z_2), \end{aligned} \tag{27}$$

where $r_{1,2}(x)$ are also formal power series, and in (27) they depend on single “resonant” variable $z_1^6 z_2$. Coefficients of these series are rational functions of system parameters a_0, a_1, b_0, b_1 and b again. The denominator of each of these functions is proportional to some integer degree $l(n)$ of the polynomial $(2 - 3b^2)$. Their numerators are polynomials in parameters of the system

$$r_{1,2}(x) = \sum_{n=1}^{\infty} \frac{q_{1,2;n}(b, a_0, a_1, b_0, b_1)}{(2 - 3b^2)^{l(n)}} x^n.$$

The condition A for the equation (27) is $6r_1(x) + r_2(x) = 0$. It is equivalent to the infinite polynomial system of equations

$$6q_{1,n}(b, a_0, a_1, b_0, b_1) + q_{2,n}(b, a_0, a_1, b_0, b_1) = 0, \quad n = 7, 14, \dots \tag{28}$$

We computed polynomials $q_{1,7}, q_{2,7}$ and solved the lowest equation from the set (28) for the parameters of solution (26). We have found 5 different two-parameter (b and a_1) solutions. With the (26) they are

$$\begin{aligned} 1) \quad & b_1 = -2a_1, \quad a_0 = a_1b, \quad b_0 = b_1b, \quad b^2 \neq 2/3, \\ 2) \quad & b_1 = (3/2)a_1, \quad a_0 = a_1b, \quad b_0 = b_1b, \quad b^2 \neq 2/3, \\ 3) \quad & b_1 = (8/3)a_1, \quad a_0 = a_1b, \quad b_0 = b_1b, \quad b^2 \neq 2/3 \end{aligned} \tag{29}$$

and

$$\begin{aligned} 4) \quad & b_1 = \frac{197-7\sqrt{745}}{24}a_1, \quad a_0 = a_1b, \quad b_0 = b_1b, \quad b^2 \neq 2/3, \\ 5) \quad & b_1 = \frac{197+7\sqrt{745}}{24}a_1, \quad a_0 = a_1b, \quad b_0 = b_1b, \quad b^2 \neq 2/3. \end{aligned} \tag{30}$$

We verified (28) up to $n = 49$ for solutions (29) for $b = 1$ and $b = 2$ and arbitrary a_1 . They are correct. We verified solution (25) in the same way. It is also correct.

Solutions (30) starting from the order $n = 14$ are correct only for the additional condition $a_1 = 0$. But for this condition, solutions (30) are a special case of solutions (29). So in accordance with the main supposition we can formulate the

Corollary 1. *If $b^2 \neq 2/3$, equalities (25) and (29) form a complete set of necessary and sufficient conditions of local integrability of system (20) at all stationary points on the manifold $v = 0$ and thus at the corresponding values of parameters (25) and (29), then system (18) is locally integrable near the degenerate point $x = y = 0$.*

Of course we should prove corollary (1) by methods independent of the “experimental mathematics”. At the moment, we can do it strictly analytically for all cases above at all stationary points except of the cases (29) 2) and 3) at stationary points $u = -3b/2, v = 0$. I.e., now we have not proved yet local integrability

by other strong methods only for cases $a_0 = a_1 b$, $b_0 = b_1 b$, $b_1 = 3a_1/2$ and $a_0 = a_1 b$, $b_0 = b_1 b$, $b_1 = 8a_1/3$ at stationary point $u = -3b/2, v = 0$. All other cases have been proved. But we strongly believe that we can finish proving for these two cases also.

4 Remark

It is interesting that for parameters of solution 2) from (29), the normal form at both stationary points is linear at least up to 49th order, i.e. $g_{1,n} = g_{2,n} = r_{1,n} = r_{2,n} = 0$, $n = 1, 2, \dots, 49$ at $b = 1$ and $b = 2$. Thus, for these values of parameters, the normalizing transformation splits the system into two independent equations. It seems that this is the simplest variant of equations (20). In the coordinates u, v the equation for the derivative of u does not depend on the variable v . The solver "DSolve" of the MATHEMATICA system gives for parameters values 2) from (29) a solution with two arbitrary constants in a finite form.

5 Conclusion

We found the complete set of necessary and sufficient conditions on parameters of system (18) for which this system is locally integrable near the degenerate stationary point $x = y = 0$. We exclude from the analysis the point $b^2 = 2/3$.

References

1. Bruno, A.D., Edneral, V.F.: Algorithmic analysis of local integrability. Dokl. Akademii Nauk 424(3), 299–303 (2009) (in Russian); Doklady Mathem. 79(1), 48–52 (2009) (in English)
2. Bruno, A.D.: Power Geometry in Algebraic and Differential Equations. Fizmatlit, Moscow (1998) (in Russian); Elsevier Science, Amsterdam (2000) (in English)
3. Bruno, A.D.: Local Methods in Nonlinear Differential Equations, Nauka, Moscow (1979) (in Russian); Springer, Berlin (1989) (in English)
4. Bruno, A.D.: Analytical form of differential equations (I,II). Trudy Moskov. Mat. Obsc. 25, 119–262 (1971); 26, 199–239 (1972) (in Russian); Trans. Moscow Math. Soc. 25, 131–288; 26, 199–239 (1972) (in English)
5. Algaba, A., Gamero, E., Garcia, C.: The integrability problem for a class of planar systems. Nonlinearity 22, 395–420 (2009)
6. Siegel, C.L.: Vorlesungen über Himmelsmechanik. Springer, Berlin (1956) (in German); Fizmatlit, Moscow (1959) (in Russian)
7. Edneral, V.F.: On algorithm of the normal form building. In: Ganzha, et al. (eds.) Proc. CASC 2007. LNCS, vol. 4770, pp. 134–142. Springer, Heidelberg (2007)

Conditions of D-Stability of the Fifth-Order Matrices

Larisa A. Burlakova

Institute for System Dynamics and Control Theory, Siberian Branch, Russian Academy of Sciences, 134, Lermontov st., Irkutsk, 664033, Russia

Abstract. The necessary conditions and some sufficient conditions represented in terms of matrix elements have been obtained for the property of D-stability of the fifth-order matrices.

1 Introduction

The concept of D-stability of matrices has appeared rather long ago, for the first time in publications on mathematical economics [1]. It has later been applied in mathematical ecology [2]. The problem is reduced to verification of positiveness for the real polynomial of n variables everywhere in the positive orthant. Only some necessary and some sufficient conditions are known for the $n \times n$ -matrices of general form ([3], [4], [5], etc.). In the general case, the condition of D-stability can hardly be verified within a finite number of steps (constructively – according to the definition given in [6]) with the use of algorithms of elimination of variables from polynomial problems because the complexity of algorithms employed for elimination of variables in polynomial optimization problems does not allow one to apply them to rather complex polynomials having large numbers and high powers of the variables [7]. So, the problem of constructing analytically verifiable conditions in terms of matrix elements is important. Necessary and sufficient conditions of D-stability for the 2nd- and 3rd-order matrices have been known for a long time ([5], [8]). In [6], one can find a general approach to solving the problem for the case of 4th-order matrices on the basis of applying the Routh–Hurwitz criterion. An algorithm of polynomial programming is used for the purpose of numerical verification of sufficient conditions and necessary conditions of D-stability for arbitrary 4×4 -matrices. This algorithm has been implemented in the form of the software applied to several particular cases when the matrix has not less than two zeros on the main diagonal. The analytical conditions for the 4th-order matrices are discussed in [9].

1.1 Some Definitions

Let $M_n(R)$ be a set of quadratic $n \times n$ -matrices over the domain R of real numbers; $\sigma(A)$ be the spectrum of matrix $A \in M_n(R)$; $D_n \subset M_n(R)$ be a class of diagonal matrices with positive elements on the main diagonal.

Definition 1. *Matrix A is called stable if $Re(\lambda) < 0$ for any $\lambda \in \sigma(A)$.*

Matrix A is stable if and only if the Routh–Hurwitz conditions hold for the characteristic polynomial

$$\det(\lambda I - A) = \lambda^n + a_1\lambda^{n-1} + \dots + a_{n-1}\lambda + a_n, \quad (1)$$

where I is the identity matrix;

$$a_j = (-1)^j \sum_{1 \leq i_1 < \dots < i_{n-j} \leq n} A_{j i_1, \dots, i_{n-j}};$$

$A_{j i_1, \dots, i_{n-j}}$ are main minors having the order j ; furthermore, $a_1 = -\text{Tr}A$, $a_n = (-1)^n \det A$. From now on, the first index in designations of the main minors denotes the minor’s order, the numbers of deleted rows and columns of matrix A are indicated in the sub-indices in the order of growth; the main diagonal minors of order k are denoted by Δ_{kk} .

Definition 2. Matrix $A \in M_n(R)$ is called *D-stable* if $\text{Re}(\lambda) < 0$ for all $\lambda \in \sigma(DA)$ for any $D \in D_n$.

Definition 3. Matrix $B \in M_n(R)$ belongs to the class P_0 if all the main minors of matrix B are non-negative, and for each $k \leq n$ there exists a strongly positive minor of matrix B , which has order k [3].

Let $X \in M_n(R)$. Let us now define the set $(-X) = \{-x \mid x \in X\}$. If matrix A of any order is D-stable then $A \in (-P_0)$ [10]. This condition, which is both necessary and sufficient for the positiveness of all the coefficients of the characteristic polynomial of matrix DA for any $d_i > 0$, requires that (i) all the main odd-order minors be non-positive (at least one minor for each odd order be strongly negative); (ii) all the main even-order minors be non-negative (at least one minor for each even order be strongly positive). Condition $A \in (-P_0)$ for the second order matrices is the necessary and sufficient condition of D-stability [10].

The following theorem is known for the 3-rd order matrices $A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{2,3} \\ a_{2,1} & a_{1,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$.

Theorem 1. [8]. Matrix $A \in M_3(R)$ is D-stable if and only if the following conditions hold: $A \in (-P_0)$; $\left(\sqrt{-A_{2,3}a_{3,3}} + \sqrt{-a_{2,2}A_{2,2}} + \sqrt{-a_{1,1}A_{2,1}}\right)^2 \geq -\Delta_{33}$, furthermore, the equality is reached only when at least one factor under the radical vanishes, and the other is nonzero.

The system of necessary inequations $A \in (-P_0)$ for the matrix $A \in M_4(R)$ is complemented by the following conditions [9]

$$\begin{aligned} \Delta_{44} &> 0, \\ 0 \leq a_{1,1}^2 \Delta_{44} &\leq -\left(\sqrt{-A_{3,3}A_{2,2,4}} + \sqrt{-A_{3,2} \Delta_{22}} + \sqrt{-A_{2,2,3} \Delta_{33}}\right)^2 a_{1,1}, \\ 0 \leq a_{2,2}^2 \Delta_{44} &\leq -\left(\sqrt{-A_{3,3}A_{2,1,4}} + \sqrt{-A_{3,1} \Delta_{22}} + \sqrt{-A_{2,1,3} \Delta_{33}}\right)^2 a_{2,2}, \\ 0 \leq a_{3,3}^2 \Delta_{44} &\leq -\left(\sqrt{-A_{3,2}A_{2,1,4}} + \sqrt{-A_{3,1}A_{2,2,4}} + \sqrt{-A_{2,1,2} \Delta_{33}}\right)^2 a_{3,3}, \\ 0 \leq a_{4,4}^2 \Delta_{44} &\leq -\left(\sqrt{-A_{3,3}A_{2,1,2}} + \sqrt{-A_{3,2}A_{2,1,3}} + \sqrt{-A_{3,1}A_{2,2,3}}\right)^2 a_{4,4}; \end{aligned} \quad (2)$$

$$\begin{aligned}
 0 \leq (-A_{31}) &\leq \left(\sqrt{-A_{21,2}a_{2,2}} + \sqrt{-A_{21,3}a_{3,3}} + \sqrt{-A_{21,4}a_{4,4}} \right)^2, \\
 0 \leq (-A_{32}) &\leq \left(\sqrt{-A_{21,2}a_{1,1}} + \sqrt{-A_{22,3}a_{3,3}} + \sqrt{-A_{22,4}a_{4,4}} \right)^2, \\
 0 \leq (-A_{33}) &\leq \left(\sqrt{A_{21,3}(-a_{1,1})} + \sqrt{A_{22,3}(-a_{2,2})} + \sqrt{\Delta_{22}(-a_{4,4})} \right)^2, \\
 0 \leq (-\Delta_{33}) &\leq \left(\sqrt{A_{21,4}(-a_{1,1})} + \sqrt{A_{22,4}(-a_{2,2})} + \sqrt{\Delta_{22}(-a_{3,3})} \right)^2;
 \end{aligned} \tag{3}$$

Furthermore, the simultaneous equality on the left in each of the groups of conditions (2) and (3) is not permitted. Some sufficient conditions of D-stability for the 4×4 matrices have been obtained in [9].

2 The 5th-Order Matrix

For the purpose of investigation of D-stability conditions for the 5th-order matrix

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} \end{pmatrix} \tag{4}$$

let us write down the characteristic equation of matrix $DA \in M_5(R)$:

$$\begin{aligned}
 &\lambda^5 + \lambda^3 (A_{23,4,5}d_1d_2 + A_{22,4,5}d_1d_3 + A_{21,4,5}d_2d_3 + A_{22,3,5}d_1d_4 + A_{21,3,5}d_2d_4 \\
 &+ A_{21,2,5}d_3d_4 + A_{22,3,4}d_1d_5 + A_{21,3,4}d_2d_5 + A_{21,2,4}d_3d_5 + A_{21,2,3}d_4d_5) \\
 &+ \lambda^2 (-A_{34,5}d_1d_2d_3 - A_{33,5}d_1d_2d_4 - A_{32,5}d_1d_3d_4 - A_{31,5}d_2d_3d_4 - A_{33,4}d_1d_2d_5 \\
 &- A_{32,4}d_1d_3d_5 - A_{31,4}d_2d_3d_5 - A_{32,3}d_1d_4d_5 - A_{31,3}d_2d_4d_5 - A_{31,2}d_3d_4d_5) \\
 &+ \lambda (A_{45}d_1d_2d_3d_4 + A_{44}d_1d_2d_3d_5 + A_{43}d_1d_2d_4d_5 + A_{42}d_1d_3d_4d_5 + A_{41}d_2d_3d_4d_5) \\
 &- d_1d_2d_3d_4d_5\Delta_5 + \lambda^4 (-d_1a_{1,1} - d_2a_{2,2} - d_3a_{3,3} - d_4a_{4,4} - d_5a_{5,5}) = 0.
 \end{aligned} \tag{5}$$

In accordance with Definition 2, matrix (4) possesses the property of D-stability if for any $D \in D_5$ the Routh–Hurwitz conditions for the characteristic polynomial (5) are satisfied. In the variant under scrutiny it is necessary to find out the fact of positiveness of polynomial (5) coefficients and positiveness of Hurwitz 2nd- and 4th-order determinants. Satisfaction of the requirement that $A \in (-P_0)$ provides for positiveness of the coefficients in the characteristic polynomial for any $d_i > 0$. The Hurwitz determinants represent polynomials of five d_i . To the end of obtaining conditions of positiveness of these polynomials let us use the principal idea of the method of quasi-homogeneous polynomial forms [7]: in the scrutinized polynomial we group simpler polynomials representing sums of some part of its terms, whose non-negativity forms the necessary conditions of positiveness of the initial polynomial. The necessary conditions of positiveness of the polynomials in the positive orthant are represented by the requirements of non-negativity of the polynomial’s coefficients with the highest and zero powers of the variable.

2.1 The Second-Order Hurwitz Determinant

The second-order Hurwitz determinant $\Gamma_2 = a_1 a_2 - a_0 a_3$ may be written in the following form:

$$\Gamma_2 = \sum d_i^2 (A_{2_{k,m,n}} d_j) (-a_{i,i}) + d_i d_j d_k (A_{3_{m,n}} - A_{2_{i,m,n}} a_{i,i} - A_{2_{m,j,n}} a_{j,j} - A_{2_{m,n,k}} a_{k,k}), \quad i \neq j \neq k \neq m \neq n, \quad i+j+k+m+n = 15; \quad i, j, k, m, n = \overline{1, 5} \quad (6)$$

(from now on the indices are placed in the order of increase). The form (6) has a total power for all d_i , i.e. the degree of the polynomial, which is 3, and with respect to each d_i – not more than 2. As obvious from the form (6), the coefficients with second powers of any variable are non-negative (the coefficient is positive at least with respect to one of the variables) when matrix (4) is such that $A \in (-P_0)$. When the following conditions

$$(A_{3_{m,n}} - A_{2_{i,m,n}} a_{i,i} - A_{2_{m,j,n}} a_{j,j} - A_{2_{m,n,k}} a_{k,k}) \geq 0, \quad i \neq j \neq k \neq m \neq n, \quad i+j+k+m+n = 15; \quad i, j, k, m, n = \overline{1, 5}, \quad (7)$$

are satisfied in addition, polynomial Γ_2 (6) is positive for any $d_i > 0$. As we intend to demonstrate below, conditions (7) are far from necessary ones.

Introduce the following denotations:

$$\begin{aligned} \beta_{12} &= A_{3_{1,2}} + (\sqrt{-A_{2_{1,2,3}} a_{3,3}} + \sqrt{-A_{2_{1,2,4}} a_{4,4}} + \sqrt{-A_{2_{1,2,5}} a_{5,5}})^2, \\ \beta_{13} &= A_{3_{1,3}} + (\sqrt{-A_{2_{1,2,3}} a_{2,2}} + \sqrt{-A_{2_{1,3,4}} a_{4,4}} + \sqrt{-A_{2_{1,3,5}} a_{5,5}})^2, \\ \beta_{14} &= A_{3_{1,4}} + (\sqrt{-A_{2_{1,2,4}} a_{2,2}} + \sqrt{-A_{2_{1,3,4}} a_{3,3}} + \sqrt{-A_{2_{1,4,5}} a_{5,5}})^2, \\ \beta_{15} &= A_{3_{1,5}} + (\sqrt{-A_{2_{1,2,5}} a_{2,2}} + \sqrt{-A_{2_{1,3,5}} a_{3,3}} + \sqrt{-A_{2_{1,4,5}} a_{4,4}})^2, \\ \beta_{23} &= A_{3_{2,3}} + (\sqrt{-A_{2_{1,2,3}} a_{1,1}} + \sqrt{-A_{2_{2,3,4}} a_{4,4}} + \sqrt{-A_{2_{2,3,5}} a_{5,5}})^2, \\ \beta_{24} &= A_{3_{2,4}} + (\sqrt{-A_{2_{1,2,4}} a_{1,1}} + \sqrt{-A_{2_{2,3,4}} a_{3,3}} + \sqrt{-A_{2_{2,4,5}} a_{5,5}})^2, \\ \beta_{25} &= A_{3_{2,5}} + (\sqrt{-A_{2_{1,2,5}} a_{1,1}} + \sqrt{-A_{2_{2,3,5}} a_{3,3}} + \sqrt{-A_{2_{2,4,5}} a_{4,4}})^2, \\ \beta_{34} &= A_{3_{3,4}} + (\sqrt{-A_{2_{1,3,4}} a_{1,1}} + \sqrt{-A_{2_{2,3,4}} a_{2,2}} + \sqrt{-A_{2_{3,4,5}} a_{5,5}})^2, \\ \beta_{35} &= A_{3_{3,5}} + (\sqrt{-A_{2_{1,3,5}} a_{1,1}} + \sqrt{-A_{2_{2,3,5}} a_{2,2}} + \sqrt{-A_{2_{3,4,5}} a_{4,4}})^2, \\ \beta_{45} &= A_{3_{4,5}} + (\sqrt{-A_{2_{1,4,5}} a_{1,1}} + \sqrt{-A_{2_{2,4,5}} a_{2,2}} + \sqrt{-A_{2_{3,4,5}} a_{3,3}})^2. \end{aligned} \quad (8)$$

Now execute necessary transformations and reduce Γ_2 (6) to the form

$$\begin{aligned} \Gamma_2 &= 1/3 \left(\sum_{i,j} d_k d_m d_n (2(A_{3_{i,j}} - A_{2_{i,j,k}} a_{k,k} - A_{2_{i,j,m}} a_{m,m} - A_{2_{i,j,n}} a_{n,n}) \right. \\ &\quad \left. + \beta_{i,j}) + \sum_i d_i \left(\sum_{j,k} (d_j \sqrt{-A_{2_{k,m,n}} a_{j,j}} - d_k \sqrt{-A_{2_{j,m,n}} a_{k,k}})^2 \right) \right), \quad (9) \\ &\quad (i \neq j \neq k; \quad i, j, k = \overline{1, 5}). \end{aligned}$$

In the form (9), for the purpose of positiveness of the expression it is sufficient to require satisfaction of the inequalities:

$$\left(2 \left(A_{3_{i,j}} - \sum_k A_{2_{i,j,k}} a_{k,k} \right) + \beta_{i,j} \right) \geq 0, \quad (i \neq j \neq k; \quad i, j, k = \overline{1, 5}). \quad (10)$$

Sufficient conditions (10) are softer than conditions (7), if in (8) $\beta_{ij} \geq 0$.

2.2 The Fourth-Order Hurwitz Determinant

The fourth-order Hurwitz determinant for the 5th-order matrix DA

$$\Gamma_4 = a_1 a_2 a_3 a_4 - a_0 a_3^2 a_4 - a_1^2 a_4^2 - a_1 a_2^2 a_5 + a_0 a_2 a_3 a_5 + 2a_0 a_1 a_4 a_5 - a_0^2 a_5^2$$

represents a homogeneous polynomial of five variables d_i having the total degree of 10, with respect to each of the variables d_i the power is not higher than 4. The polynomial contains 291 addends:

$$\Gamma_4 = \sum \left(b_{1_{i_1} 2_{i_2} 3_{i_3} 4_{i_4} 5_{i_5}} d_1^{i_1} d_2^{i_2} d_3^{i_3} d_4^{i_4} d_5^{i_5} + b_{k_{j_k} m_{j_m} n_{j_n} s_{j_s}} d_k^{j_k} d_m^{j_m} d_n^{j_n} d_s^{j_s} \right),$$

$$k < m < n < s; k, m, n, s = \overline{1, 5}; i_1 + i_2 + i_3 + i_4 + i_5 = 10, j_k + j_m + j_n + j_s = 10,$$

where denotations for polynomial coefficients introduced are such that numbers d_i are indicated in the indices, while the powers of respective d_i are given in the sub-indices. The values of these coefficients are given in the Appendix, but since expressions are bulky, the coefficients have been decomposed into groups, and in each of the groups are written only some of the coefficients. Group (A.1) includes the coefficients, which contain 4 indices, the sub-indices are all unequal and assume the values from the set 1,2,3,4; each coefficient is equal to the product of the main minors of orders 1,2,3,4; the list (A.1) contains 120 elements. If conditions (7) are satisfied then coefficients of group (A.2) are non-negative.

Noteworthy, coefficients of group (A.1) are non-negative due to the requirement that $A \in (-P_0)$. This allows one to execute some transpositions and transform the polynomial to the form

$$\begin{aligned} \Gamma_4 = & f_0 + d_1^4 (d_2^3 f_{12} + d_3^3 f_{13} + d_4^3 f_{14} + f_{15} d_5^3 + f_1) + d_2^4 (d_1^3 f_{21} + d_3^3 f_{23} \\ & + d_4^3 f_{24} + f_{25} d_5^3 + f_2) + d_3^4 (d_1^3 f_{31} + d_2^3 f_{32} + d_4^3 f_{34} + f_{35} d_5^3 + f_3) + d_4^4 (d_1^3 f_{41} \\ & + d_2^3 f_{42} + d_3^3 f_{43} + f_{45} d_5^3 + f_4) + d_5^4 (d_1^3 f_{51} + d_2^3 f_{52} + d_3^3 f_{53} + f_{54} d_4^3 + f_5), \end{aligned} \quad (11)$$

where f_0 does not contain d_i of the 4th power; f_j ($j = \overline{1, 5}$) does not contain d_j , and all the rest of d_k enter the polynomial, while having the power not higher than 2:

$$\begin{aligned} f_0 = & \sum b_{i_3 j_3 k_2 m_2} d_i^3 d_j^3 d_k^2 d_m^2 + \sum b_{i_3 j_3 k_3 m_1} d_i^3 d_j^3 d_k^3 d_m + \sum b_{i_3 j_3 k_2 m_1 n_1} d_i^3 d_j^3 d_k^2 d_m d_n \\ & + \sum b_{i_3 j_2 k_2 m_2 n_1} d_i^3 d_j^2 d_k^2 d_m^2 d_n + b_{1_2 2_2 3_2 4_2 5_2} d_1^2 d_2^2 d_3^2 d_4^2 d_5^2, \\ & (i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1, 5}). \end{aligned} \quad (12)$$

$$\begin{aligned} f_1 = & b_{1_4 2_2 3_2 4_2} d_2^2 d_3^2 d_4^2 + b_{1_4 2_2 3_2 4_1 5_1} d_2^2 d_3^2 d_4 d_5 + b_{1_4 2_2 3_1 4_2 5_1} d_2^2 d_3 d_4^2 d_5 \\ & + b_{1_4 2_1 3_2 4_2 5_1} d_2 d_3^2 d_4^2 d_5 + b_{1_4 2_2 3_2 5_2} d_2^2 d_3^2 d_5^2 + b_{1_4 2_2 3_1 4_1 5_2} d_2^2 d_3 d_4 d_5^2 \\ & + b_{1_4 2_1 3_2 4_1 5_2} d_2 d_3^2 d_4 d_5^2 + b_{1_4 2_2 4_2 5_2} d_2^2 d_4^2 d_5^2 \\ & + b_{1_4 2_1 3_1 4_2 5_2} d_2 d_3 d_4^2 d_5^2 + b_{1_4 3_2 4_2 5_2} d_3^2 d_4^2 d_5^2; \end{aligned} \quad (13)$$

$$\begin{aligned} \{ f_{12} = & d_4 \left(d_3 \sqrt{b_{1_4 2_3 3_2 4_1}} - d_5 \sqrt{b_{1_4 2_3 4_1 5_2}} \right)^2 + d_3 \left(d_4 \sqrt{b_{1_4 2_3 3_1 4_2}} - d_5 \sqrt{b_{1_4 2_3 3_1 5_2}} \right)^2 \\ & + d_5 \left(d_3 \sqrt{b_{1_4 2_3 3_2 5_1}} - d_4 \sqrt{b_{1_4 2_3 4_2 5_1}} \right)^2 + \mu_{21} d_3 d_4 d_5; \end{aligned}$$

$$\begin{aligned}
f_{13} &= d_4 \left(d_2 \sqrt{b_{1_4 2_2 3_3 4_1}} - d_5 \sqrt{b_{1_4 3_3 4_1 5_2}} \right)^2 + d_2 \left(d_4 \sqrt{b_{1_4 2_1 3_3 4_2}} - d_5 \sqrt{b_{1_4 2_1 3_3 5_2}} \right)^2 \\
&\quad + d_5 \left(d_2 \sqrt{b_{1_4 2_2 3_3 5_1}} - d_4 \sqrt{b_{1_4 3_3 4_2 5_1}} \right)^2 + \mu_{31} d_2 d_4 d_5; \\
f_{14} &= d_3 \left(\sqrt{b_{1_4 2_2 3_1 4_3}} d_2 - \sqrt{b_{1_4 3_1 4_3 5_2}} d_5 \right)^2 + d_2 \left(\sqrt{b_{1_4 2_1 3_2 4_3}} d_3 - \sqrt{b_{1_4 2_1 4_3 5_2}} d_5 \right)^2 \\
&\quad + d_5 \left(\sqrt{b_{1_4 2_2 4_3 5_1}} d_2 - \sqrt{b_{1_4 3_2 4_3 5_1}} d_3 \right)^2 + \mu_{41} d_2 d_3 d_5; \\
f_{15} &= d_3 \left(\sqrt{b_{1_4 2_2 3_1 5_3}} d_2 - \sqrt{b_{1_4 3_1 4_2 5_3}} d_4 \right)^2 + d_2 \left(\sqrt{b_{1_4 2_1 3_2 5_3}} d_3 - \sqrt{b_{1_4 2_1 4_2 5_3}} d_4 \right)^2 \\
&\quad + d_4 \left(\sqrt{b_{1_4 2_2 4_1 5_3}} d_2 - \sqrt{b_{1_4 3_2 4_1 5_3}} d_3 \right)^2 + \mu_{51} d_2 d_3 d_4 \}.
\end{aligned} \tag{14}$$

Other values of f_j and f_{jk} may be obtained from (13)–(14) by the corresponding transposition of indices and sub-indices. The values of μ_{ij} are given in the Appendix (A.3).

In the general case, it is possible to equate to zero all the three brackets simultaneously in each of the coefficients f_{ij} with d_i^4 by choosing $d_j > 0$. So, the requirements of $\mu_{ij} \geq 0$ are necessary for the positiveness of Γ_4 for any $d_j > 0$. There are 20 such conditions:

$$\begin{aligned}
\mu_{ij} &= \left(-A_{2_{k,m,n}} a_{j,j} \right) \delta_{ij} \geq 0, \\
(k < m < n; i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1,5}; \delta_{ij} = \delta_{ji}),
\end{aligned}$$

but these are reduced to ten inequalities of the form:

$$\begin{aligned}
\{ &\delta_{12} = \left(\sqrt{-A_{4_5} A_{3_{3,4}}} + \sqrt{-A_{4_4} A_{3_{3,5}}} + \sqrt{-A_{4_3} A_{3_{4,5}}} \right)^2 + A_{2_{3,4,5}} \Delta_5 \geq 0, \\
&\delta_{13} = \left(\sqrt{-A_{4_5} A_{3_{2,4}}} + \sqrt{-A_{4_4} A_{3_{2,5}}} + \sqrt{-A_{4_2} A_{3_{4,5}}} \right)^2 + A_{2_{2,4,5}} \Delta_5 \geq 0, \\
&\delta_{14} = \left(\sqrt{-A_{4_5} A_{3_{2,3}}} + \sqrt{-A_{4_3} A_{3_{2,5}}} + \sqrt{-A_{4_2} A_{3_{3,5}}} \right)^2 + A_{2_{2,3,5}} \Delta_5 \geq 0, \\
&\delta_{15} = \left(\sqrt{-A_{4_4} A_{3_{2,3}}} + \sqrt{-A_{4_3} A_{3_{2,4}}} + \sqrt{-A_{4_2} A_{3_{3,4}}} \right)^2 + A_{2_{2,3,4}} \Delta_5 \geq 0, \\
&\delta_{23} = \left(\sqrt{-A_{4_5} A_{3_{1,4}}} + \sqrt{-A_{4_4} A_{3_{1,5}}} + \sqrt{-A_{4_1} A_{3_{4,5}}} \right)^2 + A_{2_{1,4,5}} \Delta_5 \geq 0, \\
&\delta_{24} = \left(\sqrt{-A_{4_5} A_{3_{1,3}}} + \sqrt{-A_{4_3} A_{3_{1,5}}} + \sqrt{-A_{4_1} A_{3_{3,5}}} \right)^2 + A_{2_{1,3,5}} \Delta_5 \geq 0, \\
&\delta_{25} = \left(\sqrt{-A_{4_4} A_{3_{1,3}}} + \sqrt{-A_{4_3} A_{3_{1,4}}} + \sqrt{-A_{4_1} A_{3_{3,4}}} \right)^2 + A_{2_{1,3,4}} \Delta_5 \geq 0, \\
&\delta_{34} = \left(\sqrt{-A_{4_5} A_{3_{1,2}}} + \sqrt{-A_{4_2} A_{3_{1,5}}} + \sqrt{-A_{4_1} A_{3_{2,5}}} \right)^2 + A_{2_{1,2,5}} \Delta_5 \geq 0, \\
&\delta_{35} = \left(\sqrt{-A_{4_4} A_{3_{1,2}}} + \sqrt{-A_{4_2} A_{3_{1,4}}} + \sqrt{-A_{4_1} A_{3_{2,4}}} \right)^2 + A_{2_{1,2,4}} \Delta_5 \geq 0, \\
&\delta_{45} = \left(\sqrt{-A_{4_3} A_{3_{1,2}}} + \sqrt{-A_{4_2} A_{3_{1,3}}} + \sqrt{-A_{4_1} A_{3_{2,3}}} \right)^2 + A_{2_{1,2,3}} \Delta_5 \geq 0 \}.
\end{aligned} \tag{15}$$

If one considers Γ_4 (11) as a polynomial with respect to any d_i , whose coefficients represent the polynomials with respect to the rest of the variables d_j , ($i \neq j = \overline{1,5}$), then it is necessary also to provide for the non-negativity of the coefficients with the zero power of d_j in the coefficient with d_i^4 and in the addend, which does not contain variable d_i . So, in the capacity of the necessary conditions of

positiveness of Γ_4 in the positive orthant, we obtain additional two groups of inequalities, which – in terms of the main minors – write:

$$\beta_{ij} \geq 0, \quad (i, j = \overline{1, 5}, \beta_{ij} = \beta_{ji}) \tag{16}$$

$$\begin{aligned} \gamma_{ij} = & \left(\sqrt{-A_{3n,i}A_{2k,m,i}} + \sqrt{-A_{3k,i}A_{2m,n,i}} + \sqrt{-A_{3m,i}A_{2k,n,i}} \right)^2 + A_{4i}a_{j,j} \geq 0, \\ & (i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1, 5}). \end{aligned} \tag{17}$$

The list (17) contains 20 elements. When comparing inequalities (16), (17) and the necessary conditions (2), (3) known for 3rd- and 4th-order matrices [8], [9] note that these coincide with the accuracy up to the boundary. The following theorem represents the result of our above reasoning.

Theorem 2. *If matrix $A \in M_5(R)$ is D -stable then $A \in (-P_0)$, and the following conditions hold:*

$$\begin{aligned} & \left(\sqrt{-A_{3n,i}A_{2k,m,i}} + \sqrt{-A_{3k,i}A_{2m,n,i}} + \sqrt{-A_{3m,i}A_{2k,n,i}} \right)^2 + A_{4i}a_{j,j} \geq 0, \\ & \left(\sqrt{-A_{4i}A_{3j,k}} + \sqrt{-A_{4j}A_{3i,k}} + \sqrt{-A_{4k}A_{3i,j}} \right)^2 + A_{2i,j,k}\Delta_5 \geq 0, \\ & \left(\sum \sqrt{-A_{2k,i,j}a_{k,k}} \right)^2 + A_{3i,j} \geq 0, \quad (i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1, 5}) \end{aligned}$$

2.3 On the Sufficient Conditions of D-Stability

As noted above, when conditions $A \in (-P_0)$ and (7) or (10) hold, the 2-order Hurwitz determinant is positive for any matrix $D \in D_5$. Let conditions (7) be satisfied, then the necessary conditions (16) hold, and the coefficients in the group (A.2) are non-negative.

Let the following conditions for the coefficients of the group (A.8) be satisfied: $b_{i_4j_2k_2m_2} \geq 0$ ($i \neq j \neq k \neq m; i, j, k, m = \overline{1, 5}$). Hence necessary conditions (17) hold. In this case it is possible to transform coefficient f_i from (11) to the following form:

$$\begin{aligned} f_i = & 1/3 \left(d_j d_k d_m d_n \left(\sum_{j,k} \left(3b_{i_4j_2k_2m_1n_1} + 2\sqrt{b_{i_4j_2k_2m_2}} \sqrt{b_{i_4j_2k_2n_2}} \right) d_j d_k \right) \right. \\ & + \left(\sqrt{b_{i_4j_2k_2m_2}} d_j d_k d_m + \sqrt{b_{1_4j_2k_2n_2}} d_j d_k d_n - \right. \\ & \quad \left. \left. \sqrt{b_{i_4j_2m_2n_2}} d_j d_m d_n - \sqrt{b_{i_4k_2m_2n_2}} d_k d_m d_n \right) \right)^2 \\ & + \left(\sqrt{b_{i_4j_2k_2m_2}} d_j d_k d_m - \sqrt{b_{1_4j_2k_2n_2}} d_j d_k d_n + \right. \\ & \quad \left. \left. \sqrt{b_{i_4j_2m_2n_2}} d_j d_m d_n - \sqrt{b_{i_4k_2m_2n_2}} d_k d_m d_n \right) \right)^2 \\ & + \left(\sqrt{b_{i_4j_2k_2m_2}} d_j d_k d_m - \sqrt{b_{1_4j_2k_2n_2}} d_j d_k d_n - \right. \\ & \quad \left. \left. \sqrt{b_{i_4j_2m_2n_2}} d_j d_m d_n + \sqrt{b_{i_4k_2m_2n_2}} d_k d_m d_n \right) \right)^2 \Big). \end{aligned} \tag{18}$$

It follows from the representation (18) that to provide for the positiveness of coefficients with d_i^4 in (11) it is sufficient that the following inequalities (30) inequalities for (A.5) be satisfied:

$$\begin{aligned} & \left(3b_{i_4 j_2 k_2 m_1 n_1} + 2\sqrt{b_{i_4 j_2 k_2 m_2}} \sqrt{b_{i_4 j_2 k_2 n_2}} \right) \geq 0, \\ & (i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1, 5}). \end{aligned}$$

Now, to grant the positiveness of Γ_4 (11) in the positive orthant it is sufficient to have the positive addend f_0 (12).

If the coefficients (A.2) $b_{i_3 j_3 k_3 m_1} \geq 0$, then the expression $\sum b_{i_3 j_3 k_2 m_2} d_i^3 d_j^3 d_k^2 d_m^2 + \sum b_{i_3 j_3 k_3 m_1} d_i^3 d_j^3 d_k^3 d_m$ from (12) may be transformed to the form:

$$\begin{aligned} & 1/3 \sum \left(\left(d_i d_j d_k d_m \left(\sqrt{b_{i_3 j_1 k_3 m_3}} d_i - \sqrt{b_{i_1 j_3 k_3 m_3}} d_j \right)^2 d_k^2 d_m^2 \right) + \left(3b_{i_2 j_2 k_3 m_3} + \right. \right. \\ & \left. \left. 2\sqrt{b_{i_3 j_1 k_3 m_3}} \sqrt{b_{i_1 j_3 k_3 m_3}} \right) d_i^3 d_j^3 d_k^2 d_m^2 \right), \quad (i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1, 5}) \end{aligned} \quad (19)$$

It follows from (19) that in order to have non-negativity of this form, it is sufficient to require that

$$\begin{aligned} & \left(3b_{i_2 j_2 k_3 m_3} + 2\sqrt{b_{i_3 j_1 k_3 m_3}} \sqrt{b_{i_1 j_3 k_3 m_3}} \right) \geq 0, \\ & (i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1, 5}). \end{aligned} \quad (20)$$

Here we have 30 inequalities for the coefficients (A.4).

Let the coefficients (A.9) $b_{i_3 j_1 k_2 m_2 n_2} \geq 0$, then $\sum b_{i_3 j_2 k_2 m_2 n_1} d_i^3 d_j^2 d_k^2 d_m^2 d_n + b_{1_2 2_2 3_2 4_2 5_2} d_1^2 d_2^2 d_3^2 d_4^2 d_5^2$ from (12) may be written down in the form:

$$\begin{aligned} & d_1 d_2 d_3 d_4 d_5 \sum \left(\sqrt{b_{i_3 j_1 k_2 m_2 n_2}} d_i - \sqrt{b_{i_1 j_3 k_2 m_2 n_2}} d_j \right)^2 d_k d_m d_n + \left(b_{1_2 2_2 3_2 4_2 5_2} \right. \\ & \left. + 2 \sum \sqrt{b_{i_3 j_1 k_2 m_2 n_2}} \sqrt{b_{i_1 j_3 k_2 m_2 n_2}} \right) d_1^2 d_2^2 d_3^2 d_4^2 d_5^2, \\ & (i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1, 5}), \end{aligned} \quad (21)$$

and in order to grant (21) non-negativity it is sufficient that the following inequalities be satisfied:

$$\begin{aligned} & \left(b_{1_2 2_2 3_2 4_2 5_2} + 2 \sum \sqrt{b_{i_3 j_1 k_2 m_2 n_2}} \sqrt{b_{i_1 j_3 k_2 m_2 n_2}} \right) \geq 0, \\ & (i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1, 5}). \end{aligned} \quad (22)$$

When the coefficients (A.6) $b_{i_3 j_3 k_2 m_1 n_1} \geq 0$, conditions (20), (22) are satisfied, $f_0 \geq 0$ for all $D \in D_5$. Consequently, the following theorem is valid.

Theorem 3. Matrix $A \in M_5(R)$ is D-stable when the following conditions are

$$\begin{aligned} & \text{satisfied: } A \in (-P_0), \delta_{ij} \geq 0, \left(3b_{i_4 j_2 k_2 m_1 n_1} + 2\sqrt{b_{i_4 j_2 k_2 m_2}} \sqrt{b_{i_4 j_2 k_2 n_2}} \right) \geq 0, \\ & \left(3b_{i_2 j_2 k_3 m_3} + 2\sqrt{b_{i_3 j_1 k_3 m_3}} \sqrt{b_{i_1 j_3 k_3 m_3}} \right) \geq 0, b_{i_4 j_2 k_2 m_2} \geq 0, b_{i_3 j_1 k_2 m_2 n_2} \geq 0, \\ & b_{i_3 j_3 k_3 m_1} \geq 0, \left(b_{1_2 2_2 3_2 4_2 5_2} + 2 \sum \sqrt{b_{i_3 j_1 k_2 m_2 n_2}} \sqrt{b_{i_1 j_3 k_2 m_2 n_2}} \right) \geq 0, \\ & b_{i_3 j_3 k_2 m_1 n_1} \geq 0, (i \neq j \neq k \neq m \neq n; i, j, k, m, n = \overline{1, 5}). \end{aligned}$$

Note, if the coefficients (A.7) $b_{i_4 j_1 k_1 m_1 n_3} \geq 0$ then the conditions (15) be satisfied. The case, when all the inequalities turn into equalities simultaneously in the conditions of Theorem 3, requires additional investigations.

References

1. Arrow, K.J., McManus, M.: A note of dynamic stability. *Econometrica* 26, 448–454 (1958)
2. Svirezhev, Y.M., Logofet, D.O.: Stability of biological communities. Nauka, Moscow (1978) (in Russian)
3. Johnson, C.R.: Sufficient conditions for D-stability. *J. Economic Theory* 9, 53–62 (1974)
4. Johnson, C.R.: D-stability and Rreal and complex quadratic forms. *Lin. Algebra and its Applications* 9, 89–94 (1974)
5. Cross, G.W.: Three types of matrix stability. *Lin. Algebra and its Applications*. 20, 253–263 (1978)
6. Kanovei, G.V., Logofet, D.O.: D-stability of 4-by-4 matrices. *J. Comp. Math. and Math. Phys.* 38, 1429–1435 (1998)
7. Kanovei, G.V., Nephedov, V.N.: On the necessary condition of positiveness for the real polynomial of several variables in the positive orthant. *Vestnik Moskovskogo universiteta*, vol. 15(2), pp. 24–29. *Vytchislitel'naya Matematika i Kibernetika* (2000)
8. Cain, B.E.: Real, 3×3 D-stable matrices. *J. Research Nat. Bureau Standards USA B80(1)*, 75–77 (1976)
9. Burlakova, L.A.: D-Stable 4th-order matrices. *J. Sovremennye tehnologii. Systemniy analiz. Modelirovanie.* 1(21), 109–116 (2009) (in Russian)
10. Johnson, C.R.: Second, third and fourth order D-Stability. *J. Research Nat. Bureau Standards USA B78(1)*, 11–13 (1974)

Appendix

$$\begin{aligned}
 \{ & b_{1_4 3_1 4_2 5_3} = A_{4_2} A_{3_{2,3}} A_{2_{2,3,4}} a_{1,1}, \quad b_{1_4 2_1 4_2 5_3} = A_{4_3} A_{3_{2,3}} A_{2_{2,3,4}} a_{1,1}, \\
 & b_{2_4 3_1 4_2 5_3} = A_{4_1} A_{3_{1,3}} A_{2_{1,3,4}} a_{2,2}, \quad b_{1_1 2_4 4_2 5_3} = A_{4_3} A_{3_{1,3}} A_{2_{1,3,4}} a_{2,2}, \\
 & b_{2_1 3_4 4_2 5_3} = A_{4_1} A_{3_{1,2}} A_{2_{1,2,4}} a_{3,3}, \quad b_{1_1 3_4 4_2 5_3} = A_{4_2} A_{3_{1,2}} A_{2_{1,2,4}} a_{3,3}, \\
 & b_{1_3 3_1 4_4 5_2} = A_{4_2} A_{3_{2,3}} A_{2_{2,3,5}} a_{4,4}, \quad b_{1_3 2_1 4_4 5_2} = A_{4_3} A_{3_{2,3}} A_{2_{2,3,5}} a_{4,4}, \\
 & b_{2_1 3_2 4_3 5_4} = A_{4_1} A_{3_{1,2}} A_{2_{1,2,3}} a_{5,5}, \quad b_{1_1 3_2 4_3 5_4} = A_{4_2} A_{3_{1,2}} A_{2_{1,2,3}} a_{5,5}, \\
 & \dots \text{ The list contains 120 elements} \}.
 \end{aligned} \tag{A.1}$$

$$\begin{aligned}
 \{ & b_{1_3 2_3 3_3 5_1} = A_{4_4} A_{3_{4,5}} (-A_{3_{4,5}} + A_{2_{1,4,5}} a_{1,1} + A_{2_{2,4,5}} a_{2,2} + A_{2_{3,4,5}} a_{3,3}), \\
 & b_{1_3 2_1 3_3 4_3} = A_{4_5} A_{3_{2,5}} (-A_{3_{2,5}} + A_{2_{1,2,5}} a_{1,1} + A_{2_{2,3,5}} a_{3,3} + A_{2_{2,4,5}} a_{4,4}), \\
 & b_{1_1 2_3 4_3 5_3} = A_{4_3} A_{3_{1,3}} (-A_{3_{1,3}} + A_{2_{1,2,3}} a_{2,2} + A_{2_{1,3,4}} a_{4,4} + A_{2_{1,3,5}} a_{5,5}), \\
 & b_{2_3 3_1 4_3 5_3} = A_{4_1} A_{3_{1,3}} (-A_{3_{1,3}} + A_{2_{1,2,3}} a_{2,2} + A_{2_{1,3,4}} a_{4,4} + A_{2_{1,3,5}} a_{5,5}), \\
 & b_{1_1 3_3 4_3 5_3} = A_{4_2} A_{3_{1,2}} (-A_{3_{1,2}} + A_{2_{1,2,3}} a_{3,3} + A_{2_{1,2,4}} a_{4,4} + A_{2_{1,2,5}} a_{5,5}), \\
 & \dots \text{ The list contains 20 elements} \};
 \end{aligned} \tag{A.2}$$

$$\begin{aligned}
& \{ \mu_{12} = b_{1_3 2_4 3_1 4_1 5_1} + 2\sqrt{b_{1_3 2_4 3_2 5_1}} \sqrt{b_{1_3 2_4 4_2 5_1}} + 2\sqrt{b_{1_3 2_4 3_1 4_2}} \sqrt{b_{1_3 2_4 3_1 5_2}} \\
& + 2\sqrt{b_{1_3 2_4 3_2 4_1}} \sqrt{b_{1_3 2_4 4_1 5_2}}, \quad \mu_{13} = b_{1_3 2_1 3_4 4_1 5_1} + 2\sqrt{b_{1_3 2_2 3_4 5_1}} \sqrt{b_{1_3 3_4 4_2 5_1}} \\
& + 2\sqrt{b_{1_3 2_1 3_4 4_2}} \sqrt{b_{1_3 2_1 3_4 5_2}} + 2\sqrt{b_{1_3 2_2 3_4 4_1}} \sqrt{b_{1_3 3_4 4_1 5_2}}, \quad \mu_{14} = b_{1_3 2_1 3_1 4_4 5_1} \\
& + 2\sqrt{b_{1_3 2_2 4_4 5_1}} \sqrt{b_{1_3 3_2 4_4 5_1}} + 2\sqrt{b_{1_3 2_1 3_2 4_4}} \sqrt{b_{1_3 2_1 4_4 5_2}} + 2\sqrt{b_{1_3 2_2 3_1 4_4}} \sqrt{b_{1_3 3_1 4_4 5_2}}, \\
& \mu_{15} = b_{1_3 2_1 3_1 4_1 5_4} + 2\sqrt{b_{1_3 2_2 4_1 5_4}} \sqrt{b_{1_3 3_2 4_1 5_4}} + 2\sqrt{b_{1_3 2_1 3_2 5_4}} \sqrt{b_{1_3 2_1 4_2 5_4}} \\
& + 2\sqrt{b_{1_3 2_2 3_1 5_4}} \sqrt{b_{1_3 3_1 4_2 5_4}}, \dots \text{The list contains 20 elements} \}.
\end{aligned} \tag{A.3}$$

$$\begin{aligned}
& \{ b_{1_2 2_3 3_3 4_2} = A_{4_5} (-2A_{3_{1,5}} A_{3_{4,5}} + A_{3_{1,5}} A_{2_{1,4,5}} a_{1,1} + A_{3_{4,5}} A_{2_{1,4,5}} a_{4,4} \\
& + a_{2,2} (A_{3_{4,5}} A_{2_{1,2,5}} + A_{3_{2,5}} A_{2_{1,4,5}} + A_{3_{1,5}} A_{2_{2,4,5}}) + a_{3,3} (A_{3_{4,5}} A_{2_{1,3,5}} \\
& + A_{3_{3,5}} A_{2_{1,4,5}} + A_{3_{1,5}} A_{2_{3,4,5}} - 2A_{4_5} a_{2,2}), \quad b_{1_2 2_3 3_2 5_2} = A_{4_4} (-2A_{3_{3,4}} A_{3_{4,5}} \\
& + A_{3_{3,4}} A_{2_{3,4,5}} a_{3,3} + A_{3_{4,5}} A_{2_{3,4,5}} a_{5,5} + a_{1,1} (A_{3_{4,5}} A_{2_{1,3,4}} + A_{3_{3,4}} A_{2_{1,4,5}} \\
& + A_{3_{1,4}} A_{2_{3,4,5}}) + a_{2,2} (A_{3_{4,5}} A_{2_{2,3,4}} + A_{3_{3,4}} A_{2_{2,4,5}} + A_{3_{2,4}} A_{2_{3,4,5}} - 2A_{4_4} a_{1,1}), \\
& b_{1_2 2_3 4_2 5_2} = A_{4_3} (-2A_{3_{3,4}} A_{3_{3,5}} + A_{3_{3,4}} A_{2_{3,4,5}} a_{4,4} + A_{3_{3,5}} A_{2_{3,4,5}} a_{5,5} \\
& + a_{1,1} (A_{3_{3,5}} A_{2_{1,3,4}} + A_{3_{3,4}} A_{2_{1,3,5}} + A_{3_{1,3}} A_{2_{3,4,5}}) + a_{2,2} (A_{3_{3,5}} A_{2_{2,3,4}} \\
& + A_{3_{3,4}} A_{2_{2,3,5}} + A_{3_{2,3}} A_{2_{3,4,5}} - 2A_{4_3} a_{1,1}), \quad b_{1_3 3_3 4_2 5_2} = A_{4_2} (-2A_{3_{2,4}} A_{3_{2,5}} \\
& + A_{3_{2,4}} A_{2_{2,4,5}} a_{4,4} + A_{3_{2,5}} A_{2_{2,4,5}} a_{5,5} + a_{1,1} (A_{3_{2,5}} A_{2_{1,2,4}} + A_{3_{2,4}} A_{2_{1,2,5}} \\
& + A_{3_{1,2}} A_{2_{2,4,5}}) + a_{3,3} (A_{3_{2,5}} A_{2_{2,3,4}} + A_{3_{2,4}} A_{2_{2,3,5}} + A_{3_{2,3}} A_{2_{2,4,5}} - 2A_{4_2} a_{1,1}), \\
& \dots \text{The list contains 30 elements} \};
\end{aligned} \tag{A.4}$$

$$\begin{aligned}
& \{ b_{1_4 2_1 3_1 4_2 5_2} = a_{1,1} (A_{4_5} A_{3_{2,3}} A_{2_{2,3,4}} + A_{4_3} A_{3_{2,5}} A_{2_{2,3,4}} + A_{4_2} A_{3_{3,5}} A_{2_{2,3,4}} \\
& + A_{4_4} A_{3_{2,3}} A_{2_{2,3,5}} + A_{4_3} A_{3_{2,4}} A_{2_{2,3,5}} + A_{4_2} A_{3_{3,4}} A_{2_{2,3,5}} + A_{4_3} A_{3_{2,3}} A_{2_{2,4,5}} \\
& + A_{4_2} A_{3_{2,3}} A_{2_{3,4,5}} - 2A_{2_{2,3,4}} A_{2_{2,3,5}} \Delta_5 - 2A_{4_2} A_{4_3} a_{1,1}), \quad b_{1_2 2_4 3_1 4_1 5_2} = \\
& a_{2,2} (A_{4_5} A_{3_{3,4}} A_{2_{1,3,4}} + A_{4_4} A_{3_{3,5}} A_{2_{1,3,4}} + A_{4_3} A_{3_{4,5}} A_{2_{1,3,4}} + A_{4_4} A_{3_{3,4}} A_{2_{1,3,5}} \\
& + A_{4_3} A_{3_{3,4}} A_{2_{1,4,5}} + A_{4_4} A_{3_{1,3}} A_{2_{3,4,5}} + A_{4_3} A_{3_{1,4}} A_{2_{3,4,5}} + A_{4_1} A_{3_{3,4}} A_{2_{3,4,5}} \\
& - 2A_{2_{1,3,4}} A_{2_{3,4,5}} \Delta_5 - 2A_{4_3} A_{4_4} a_{2,2}), \quad b_{1_1 2_2 3_4 4_1 5_2} = a_{3,3} (A_{4_5} A_{3_{1,4}} A_{2_{1,2,4}} \\
& + A_{4_4} A_{3_{1,5}} A_{2_{1,2,4}} + A_{4_1} A_{3_{4,5}} A_{2_{1,2,4}} + A_{4_4} A_{3_{1,4}} A_{2_{1,2,5}} + A_{4_4} A_{3_{1,2}} A_{2_{1,4,5}} \\
& + A_{4_2} A_{3_{1,4}} A_{2_{1,4,5}} + A_{4_1} A_{3_{2,4}} A_{2_{1,4,5}} + A_{4_1} A_{3_{1,4}} A_{2_{2,4,5}} - 2A_{2_{1,2,4}} A_{2_{1,4,5}} \Delta_5 \\
& - 2A_{4_1} A_{4_4} a_{3,3}), \quad b_{1_2 2_1 3_1 4_4 5_2} = a_{4,4} (A_{4_5} A_{3_{2,3}} A_{2_{1,2,3}} + A_{4_3} A_{3_{2,5}} A_{2_{1,2,3}} \\
& + A_{4_2} A_{3_{3,5}} A_{2_{1,2,3}} + A_{4_3} A_{3_{2,3}} A_{2_{1,2,5}} + A_{4_2} A_{3_{2,3}} A_{2_{1,3,5}} + A_{4_3} A_{3_{1,2}} A_{2_{2,3,5}} \\
& + A_{4_2} A_{3_{1,3}} A_{2_{2,3,5}} + A_{4_1} A_{3_{2,3}} A_{2_{2,3,5}} - 2A_{2_{1,2,3}} A_{2_{2,3,5}} \Delta_5 - 2A_{4_2} A_{4_3} a_{4,4}), \\
& \dots \text{The list contains 30 elements} \};
\end{aligned} \tag{A.5}$$

$$\begin{aligned}
& \{ b_{1_1 2_2 3_3 4_3 5_1} = A_{2_{1,2,5}} (A_{4_5} A_{3_{1,3}} + A_{4_3} A_{3_{1,5}} + A_{4_1} A_{3_{3,5}} - A_{2_{1,3,5}} \Delta_5) a_{3,3} \\
& + A_{2_{1,2,5}} (A_{4_5} A_{3_{1,4}} + A_{4_4} A_{3_{1,5}} + A_{4_1} A_{3_{4,5}} - A_{2_{1,4,5}} \Delta_5) a_{4,4} - 4A_{4_1} A_{4_5} a_{3,3} a_{4,4} \\
& - (A_{4_2} A_{3_{1,5}} + A_{4_1} A_{3_{2,5}} + A_{4_5} A_{3_{1,2}} - A_{2_{1,2,5}} \Delta_5) (A_{3_{1,5}} - A_{2_{1,2,5}} a_{2,2} \\
& - A_{2_{1,3,5}} a_{3,3} - A_{2_{1,4,5}} a_{4,4}) - A_{4_1} A_{3_{1,5}} (A_{3_{2,5}} - A_{2_{1,2,5}} a_{1,1} - A_{2_{2,3,5}} a_{3,3} \\
& - A_{2_{2,4,5}} a_{4,4}) - A_{4_5} A_{3_{1,5}} (A_{3_{1,2}} - A_{2_{1,2,3}} a_{3,3} - A_{2_{1,2,4}} a_{4,4} - A_{2_{1,2,5}} a_{5,5}), \\
& \dots \text{The list contains 30 elements} \};
\end{aligned} \tag{A.6}$$

$$\begin{aligned}
\{ & b_{1_1 2_1 3_1 4_1 5_3} = A_{2_2, 3_4} a_{1, 1} (A_{4_4} A_{3_2, 3} + A_{4_3} A_{3_2, 4} + A_{4_2} A_{3_3, 4} - A_{2_2, 3, 4} \Delta_5), \\
& b_{1_1 2_1 3_1 4_1 5_3} = A_{2_1, 3_4} a_{2, 2} (A_{4_4} A_{3_1, 3} + A_{4_3} A_{3_1, 4} + A_{4_1} A_{3_3, 4} - A_{2_1, 3, 4} \Delta_5), \\
& b_{1_1 2_1 3_1 4_1 5_3} = A_{2_1, 2_4} a_{3, 3} (A_{4_4} A_{3_1, 2} + A_{4_2} A_{3_1, 4} + A_{4_1} A_{3_2, 4} - A_{2_1, 2, 4} \Delta_5), \\
& b_{1_1 2_1 3_1 4_1 5_3} = A_{2_1, 2_3} a_{4, 4} (A_{4_3} A_{3_1, 2} + A_{4_2} A_{3_1, 3} + A_{4_1} A_{3_2, 3} - A_{2_1, 2, 3} \Delta_5), \\
& b_{1_1 2_3 3_1 4_1 5_4} = A_{2_1, 3_4} a_{5, 5} (A_{4_4} A_{3_1, 3} + A_{4_3} A_{3_1, 4} + A_{4_1} A_{3_3, 4} - A_{2_1, 3, 4} \Delta_5), \\
& \dots \text{The list contains 20 elements} \};
\end{aligned} \tag{A.7}$$

$$\begin{aligned}
\{ & b_{1_1 3_2 4_2 5_2} = A_{4_2} a_{1, 1} (A_{3_2, 5} A_{2_2, 3, 4} + A_{3_2, 4} A_{2_2, 3, 5} + A_{3_2, 3} A_{2_2, 4, 5} - A_{4_2} a_{1, 1}), \\
& b_{2_4 3_2 4_2 5_2} = A_{4_1} a_{2, 2} (A_{3_1, 5} A_{2_1, 3, 4} + A_{3_1, 4} A_{2_1, 3, 5} + A_{3_1, 3} A_{2_1, 4, 5} - A_{4_1} a_{2, 2}), \\
& b_{1_2 2_2 3_2 5_2} = A_{4_4} a_{3, 3} (A_{3_4, 5} A_{2_1, 2, 4} + A_{3_2, 4} A_{2_1, 4, 5} + A_{3_1, 4} A_{2_2, 4, 5} - A_{4_4} a_{3, 3}), \\
& b_{1_2 2_2 3_2 4_4} = A_{4_5} a_{4, 4} (A_{3_3, 5} A_{2_1, 2, 5} + A_{3_2, 5} A_{2_1, 3, 5} + A_{3_1, 5} A_{2_2, 3, 5} - A_{4_5} a_{4, 4}), \\
& b_{1_2 2_2 4_2 5_4} = A_{4_3} a_{5, 5} (A_{3_3, 4} A_{2_1, 2, 3} + A_{3_2, 3} A_{2_1, 3, 4} + A_{3_1, 3} A_{2_2, 3, 4} - A_{4_3} a_{5, 5}), \\
& \dots \text{The list contains 20 elements} \};
\end{aligned} \tag{A.8}$$

$$\begin{aligned}
\{ & b_{1_3 2_2 3_1 4_2 5_2} = -A_{3_3, 5} (A_{4_4} A_{3_2, 3} + A_{4_3} A_{3_2, 4} + A_{4_2} A_{3_3, 4} - A_{2_2, 3, 4} \Delta_5) \\
& - A_{3_3, 4} (A_{4_5} A_{3_2, 3} + A_{4_3} A_{3_2, 5} + A_{4_2} A_{3_3, 5} - A_{2_2, 3, 5} \Delta_5) - A_{3_2, 3} (A_{4_5} A_{3_3, 4} \\
& + A_{4_4} A_{3_3, 5} + A_{4_3} A_{3_4, 5} - A_{2_3, 4, 5} \Delta_5) + (A_{2_3, 4, 5} (A_{4_3} A_{3_1, 2} + A_{4_2} A_{3_1, 3} + A_{4_1} A_{3_2, 3} \\
& - A_{2_1, 2, 3} \Delta_5) + A_{2_2, 3, 5} (A_{4_4} A_{3_1, 3} + A_{4_3} A_{3_1, 4} + A_{4_1} A_{3_3, 4} - A_{2_1, 3, 4} \Delta_5) \\
& + A_{2_2, 3, 4} (A_{4_5} A_{3_1, 3} + A_{4_3} A_{3_1, 5} + A_{4_1} A_{3_3, 5} - A_{2_1, 3, 5} \Delta_5) + A_{2_1, 3, 5} (A_{4_4} A_{3_2, 3} \\
& + A_{4_3} A_{3_2, 4} + A_{4_2} A_{3_3, 4} - A_{2_2, 3, 4} \Delta_5) + A_{2_1, 3, 4} (A_{4_5} A_{3_2, 3} + A_{4_3} A_{3_2, 5} + A_{4_2} A_{3_3, 5} \\
& - A_{2_2, 3, 5} \Delta_5) + A_{2_1, 2, 3} (A_{4_5} A_{3_3, 4} + A_{4_4} A_{3_3, 5} + A_{4_3} A_{3_4, 5} - A_{2_3, 4, 5} \Delta_5)) a_{1, 1} + \\
& (A_{2_2, 3, 5} (A_{4_4} A_{3_2, 3} + A_{4_3} A_{3_2, 4} + A_{4_2} A_{3_3, 4} - A_{2_2, 3, 4} \Delta_5) + A_{2_2, 3, 4} (A_{4_5} A_{3_2, 3} \\
& + A_{4_3} A_{3_2, 5} + A_{4_2} A_{3_3, 5} - A_{2_2, 3, 5} \Delta_5)) a_{2, 2} - A_{4_3} A_{3_2, 3} (A_{3_4, 5} - A_{2_1, 4, 5} a_{1, 1} \\
& - A_{2_2, 4, 5} a_{2, 2} - A_{2_3, 4, 5} a_{3, 3}) + (A_{2_3, 4, 5} (A_{4_4} A_{3_2, 3} + A_{4_3} A_{3_2, 4} + A_{4_2} A_{3_3, 4} \\
& - A_{2_2, 3, 4} \Delta_5) + A_{2_2, 3, 4} (A_{4_5} A_{3_3, 4} + A_{4_4} A_{3_3, 5} + A_{4_3} A_{3_4, 5} - A_{2_3, 4, 5} \Delta_5)) a_{4, 4} \\
& - A_{4_3} A_{3_3, 4} (A_{3_2, 5} - A_{2_1, 2, 5} a_{1, 1} - A_{2_2, 3, 5} a_{3, 3} - A_{2_2, 4, 5} a_{4, 4}) + (A_{2_3, 4, 5} (A_{4_5} A_{3_2, 3} \\
& + A_{4_3} A_{3_2, 5} + A_{4_2} A_{3_3, 5} - A_{2_2, 3, 5} \Delta_5) + A_{2_2, 3, 5} (A_{4_5} A_{3_3, 4} + A_{4_4} A_{3_3, 5} + A_{4_3} A_{3_4, 5} \\
& - A_{2_3, 4, 5} \Delta_5)) a_{5, 5} - A_{4_3} A_{3_3, 5} (A_{3_2, 4} - A_{2_1, 2, 4} a_{1, 1} - A_{2_2, 3, 4} a_{3, 3} - A_{2_2, 4, 5} a_{5, 5}) \\
& + (A_{4_3} A_{3_1, 3} A_{2_2, 4, 5} + 2A_{4_3} \Delta_5) a_{1, 1} + A_{4_5} A_{3_3, 5} A_{2_2, 3, 4} a_{5, 5} + A_{4_4} A_{3_3, 4} A_{2_2, 3, 5} a_{4, 4} \\
& - 2A_{4_1} A_{4_3} a_{1, 1}^2 + A_{4_2} A_{3_2, 3} A_{2_3, 4, 5} a_{2, 2} - 4A_{4_2} A_{4_3} a_{1, 1} a_{2, 2} - 2A_{4_3}^2 a_{1, 1} a_{3, 3} \\
& - 4A_{4_3} A_{4_4} a_{1, 1} a_{4, 4} - 4A_{4_3} A_{4_5} a_{1, 1} a_{5, 5}, \dots \text{The list contains 20 elements} \};
\end{aligned} \tag{A.9}$$

$$\begin{aligned}
& b_{1_2 2_2 3_2 4_2 5_2} = -\Delta_5^2 - A_{4_1}^2 a_{1, 1}^2 - A_{4_2}^2 a_{2, 2}^2 - A_{4_3}^2 a_{3, 3}^2 - A_{4_4}^2 a_{4, 4}^2 - A_{4_5}^2 a_{5, 5}^2 \\
& + \Delta_5 (A_{3_4, 5} A_{2_1, 2, 3} + A_{3_3, 5} A_{2_1, 2, 4} + A_{3_3, 4} A_{2_1, 2, 5} + A_{3_2, 5} A_{2_1, 3, 4} + A_{3_2, 4} A_{2_1, 3, 5} \\
& + A_{3_2, 3} A_{2_1, 4, 5} + A_{3_1, 5} A_{2_2, 3, 4} + A_{3_1, 4} A_{2_2, 3, 5} + A_{3_1, 3} A_{2_2, 4, 5} + A_{3_1, 2} A_{2_3, 4, 5} \\
& - 2(A_{2_1, 2, 5} A_{2_1, 3, 4} + A_{2_1, 2, 4} A_{2_1, 3, 5} + A_{2_1, 2, 3} A_{2_1, 4, 5}) a_{1, 1} - 2(A_{2_1, 2, 5} A_{2_2, 3, 4} \\
& + A_{2_1, 2, 4} A_{2_2, 3, 5} + A_{2_1, 2, 3} A_{2_2, 4, 5}) a_{2, 2} - 2(A_{2_1, 3, 5} A_{2_2, 3, 4} + A_{2_1, 3, 4} A_{2_2, 3, 5} \\
& + A_{2_1, 2, 3} A_{2_3, 4, 5}) a_{3, 3} - 2(A_{2_1, 4, 5} A_{2_2, 3, 4} + A_{2_1, 3, 4} A_{2_2, 4, 5} \\
& + A_{2_1, 2, 4} A_{2_3, 4, 5}) a_{4, 4} - 2(A_{2_1, 4, 5} A_{2_2, 3, 5} + A_{2_1, 3, 5} A_{2_2, 4, 5} + A_{2_1, 2, 5} A_{2_3, 4, 5}) a_{5, 5}) \\
& + A_{4_5} (-2(A_{3_1, 4} A_{3_2, 3} + A_{3_1, 3} A_{3_2, 4} + A_{3_1, 2} A_{3_3, 4}) + (A_{3_1, 4} A_{2_1, 2, 3}
\end{aligned}$$

$$\begin{aligned}
& +A_{3_{1,3}}A_{2_{1,2,4}} + A_{3_{1,2}}A_{2_{1,3,4}})a_{1,1} + (A_{3_{2,4}}A_{2_{1,2,3}} + A_{3_{2,3}}A_{2_{1,2,4}} \\
& +A_{3_{1,2}}A_{2_{2,3,4}})a_{2,2} + (A_{3_{3,4}}A_{2_{1,2,3}} + A_{3_{2,3}}A_{2_{1,3,4}} + A_{3_{1,3}}A_{2_{2,3,4}})a_{3,3} \\
& + (A_{3_{3,4}}A_{2_{1,2,4}} + A_{3_{2,4}}A_{2_{1,3,4}} + A_{3_{1,4}}A_{2_{2,3,4}})a_{4,4} + (A_{3_{4,5}}A_{2_{1,2,3}} \\
& +A_{3_{3,5}}A_{2_{1,2,4}} + A_{3_{3,4}}A_{2_{1,2,5}} + A_{3_{2,5}}A_{2_{1,3,4}} + A_{3_{2,4}}A_{2_{1,3,5}} + A_{3_{2,3}}A_{2_{1,4,5}} \\
& +A_{3_{1,5}}A_{2_{2,3,4}} + A_{3_{1,4}}A_{2_{2,3,5}} + A_{3_{1,3}}A_{2_{2,4,5}} + A_{3_{1,2}}A_{2_{3,4,5}} + 2\Delta_5)a_{5,5}) \\
& +A_{44}(-2(A_{3_{1,5}}A_{3_{2,3}} + A_{3_{1,3}}A_{3_{2,5}} + A_{3_{1,2}}A_{3_{3,5}}) + (A_{3_{1,5}}A_{2_{1,2,3}} \\
& +A_{3_{1,3}}A_{2_{1,2,5}} + A_{3_{1,2}}A_{2_{1,3,5}})a_{1,1} + (A_{3_{2,5}}A_{2_{1,2,3}} + A_{3_{2,3}}A_{2_{1,2,5}} \\
& +A_{3_{1,2}}A_{2_{2,3,5}})a_{2,2} + (A_{3_{3,5}}A_{2_{1,2,3}} + A_{3_{2,3}}A_{2_{1,3,5}} + A_{3_{1,3}}A_{2_{2,3,5}})a_{3,3} \\
& +(A_{3_{3,5}}A_{2_{1,2,5}} + A_{3_{2,5}}A_{2_{1,3,5}} + A_{3_{1,5}}A_{2_{2,3,5}})a_{5,5} + a_{4,4}(A_{3_{4,5}}A_{2_{1,2,3}} \\
& +A_{3_{3,5}}A_{2_{1,2,4}} + A_{3_{3,4}}A_{2_{1,2,5}} + A_{3_{2,5}}A_{2_{1,3,4}} + A_{3_{2,4}}A_{2_{1,3,5}} \\
& +A_{3_{2,3}}A_{2_{1,4,5}} + A_{3_{1,5}}A_{2_{2,3,4}} + A_{3_{1,4}}A_{2_{2,3,5}} + A_{3_{1,3}}A_{2_{2,4,5}} + A_{3_{1,2}}A_{2_{3,4,5}} \\
& +2\Delta_5 - 4A_{45}a_{5,5})) + A_{43}(-2(A_{3_{1,5}}A_{3_{2,4}} + A_{3_{1,4}}A_{3_{2,5}} \\
& +A_{3_{1,2}}A_{3_{4,5}}) + (A_{3_{1,5}}A_{2_{1,2,4}} + A_{3_{1,4}}A_{2_{1,2,5}} + A_{3_{1,2}}A_{2_{1,4,5}})a_{1,1} \\
& +(A_{3_{2,5}}A_{2_{1,2,4}} + A_{3_{2,4}}A_{2_{1,2,5}} + A_{3_{1,2}}A_{2_{2,4,5}})a_{2,2} + (A_{3_{4,5}}A_{2_{1,2,4}} \\
& +A_{3_{2,4}}A_{2_{1,4,5}} + A_{3_{1,4}}A_{2_{2,4,5}})a_{4,4} + (A_{3_{4,5}}A_{2_{1,2,5}} + A_{3_{2,5}}A_{2_{1,4,5}} \\
& +A_{3_{1,5}}A_{2_{2,4,5}})a_{5,5} + a_{3,3}(A_{3_{4,5}}A_{2_{1,2,3}} + A_{3_{3,5}}A_{2_{1,2,4}} + A_{3_{3,4}}A_{2_{1,2,5}} \\
& +A_{3_{2,5}}A_{2_{1,3,4}} + A_{3_{2,4}}A_{2_{1,3,5}} + A_{3_{2,3}}A_{2_{1,4,5}} + A_{3_{1,5}}A_{2_{2,3,4}} \\
& +A_{3_{1,4}}A_{2_{2,3,5}} + A_{3_{1,3}}A_{2_{2,4,5}} + A_{3_{1,2}}A_{2_{3,4,5}} + 2\Delta_5 - 4A_{44}a_{4,4} \\
& -4A_{45}a_{5,5})) + A_{42}(-2(A_{3_{1,5}}A_{3_{3,4}} + A_{3_{1,4}}A_{3_{3,5}} + A_{3_{1,3}}A_{3_{4,5}}) \\
& +(A_{3_{1,5}}A_{2_{1,3,4}} + A_{3_{1,4}}A_{2_{1,3,5}} + A_{3_{1,3}}A_{2_{1,4,5}})a_{1,1} + (A_{3_{3,5}}A_{2_{1,3,4}} \\
& +A_{3_{3,4}}A_{2_{1,3,5}} + A_{3_{1,3}}A_{2_{3,4,5}})a_{3,3} + (A_{3_{4,5}}A_{2_{1,3,4}} + A_{3_{3,4}}A_{2_{1,4,5}} \\
& +A_{3_{1,4}}A_{2_{3,4,5}})a_{4,4} + (A_{3_{4,5}}A_{2_{1,3,5}} + A_{3_{3,5}}A_{2_{1,4,5}} + A_{3_{1,5}}A_{2_{3,4,5}})a_{5,5} \\
& +a_{2,2}(A_{3_{4,5}}A_{2_{1,2,3}} + A_{3_{3,5}}A_{2_{1,2,4}} + A_{3_{3,4}}A_{2_{1,2,5}} + A_{3_{2,5}}A_{2_{1,3,4}} \\
& +A_{3_{2,4}}A_{2_{1,3,5}} + A_{3_{2,3}}A_{2_{1,4,5}} + A_{3_{1,5}}A_{2_{2,3,4}} + A_{3_{1,4}}A_{2_{2,3,5}} \\
& +A_{3_{1,3}}A_{2_{2,4,5}} + A_{3_{1,2}}A_{2_{3,4,5}} + 2\Delta_5 - 4A_{43}a_{3,3} - 4A_{44}a_{4,4} \\
& -4A_{45}a_{5,5})) + A_{41}(-2(A_{3_{2,5}}A_{3_{3,4}} + A_{3_{2,4}}A_{3_{3,5}} + A_{3_{2,3}}A_{3_{4,5}}) + (A_{3_{2,5}}A_{2_{2,3,4}} \\
& +A_{3_{2,4}}A_{2_{2,3,5}} + A_{3_{2,3}}A_{2_{2,4,5}})a_{2,2} + (A_{3_{3,5}}A_{2_{2,3,4}} + A_{3_{3,4}}A_{2_{2,3,5}} + A_{3_{2,3}}A_{2_{3,4,5}})a_{3,3} \\
& +(A_{3_{4,5}}A_{2_{2,3,4}} + A_{3_{3,4}}A_{2_{2,4,5}} + A_{3_{2,4}}A_{2_{3,4,5}})a_{4,4} + (A_{3_{4,5}}A_{2_{2,3,5}} + A_{3_{3,5}}A_{2_{2,4,5}} \\
& +A_{3_{2,5}}A_{2_{3,4,5}})a_{5,5} + a_{1,1}(A_{3_{4,5}}A_{2_{1,2,3}} + A_{3_{3,5}}A_{2_{1,2,4}} + A_{3_{3,4}}A_{2_{1,2,5}} + A_{3_{2,5}}A_{2_{1,3,4}} \\
& +A_{3_{2,4}}A_{2_{1,3,5}} + A_{3_{2,3}}A_{2_{1,4,5}} + A_{3_{1,5}}A_{2_{2,3,4}} + A_{3_{1,4}}A_{2_{2,3,5}} + A_{3_{1,3}}A_{2_{2,4,5}} \\
& +A_{3_{1,2}}A_{2_{3,4,5}} + 2\Delta_5 - 4A_{42}a_{2,2} - 4A_{43}a_{3,3} - 4A_{44}a_{4,4} - 4A_{45}a_{5,5})).
\end{aligned}$$

All the computations have been executed with the aid of the software MATHEMATICA.

Code Generation for Polynomial Multiplication

Ling Ding¹ and Éric Schost²

- ¹ ORCCA, Computer Science Department, The University of Western Ontario,
London, Ontario, Canada
lding6@csd.uwo.ca
- ² ORCCA, Computer Science Department, The University of Western Ontario,
London, Ontario, Canada
eschost@uwo.ca

Abstract. We discuss the family of “divide-and-conquer” algorithms for polynomial multiplication, that generalize Karatsuba’s algorithm. We give explicit versions of *transposed* and *short* products for this family of algorithms and describe code generation techniques that result in high-performance implementations.

1 Introduction

Polynomial multiplication is a cornerstone of higher-level algorithms: fast algorithms for Euclidean division, GCD, Chinese remaindering, factorization, Newton iteration, etc, depend on fast (subquadratic) algorithms for polynomial multiplication [20]. This article describes implementation techniques for several aspects of this question; we focus on *dense* polynomial arithmetic, as opposed the sparse model [12].

Variants of polynomial multiplication. To fix notation, we let \mathbb{R} be our base ring, and for $n \in \mathbb{N}_{>0}$, we let $\mathbb{R}[x]_n$ be the set of polynomials in $\mathbb{R}[x]$ of degree less than n . We will write the input polynomials as

$$A = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} \in \mathbb{R}[x]_n, \quad B = b_0 + b_1x + \cdots + b_{n-1}x^{n-1} \in \mathbb{R}[x]_n;$$

note that the *number of terms* of A and B is at most n . Note also that we assume the same degree upper bound on A and B : this needs not be a sensible assumption in general, but makes sense in many applications (such as Newton iteration or GCD). Our first objective is to compute the coefficients of the product

$$C = AB = c_0x + c_1x + \cdots + c_{2n-2}x^{2n-2} \in \mathbb{R}[x]_{2n-1}.$$

This operation will be called *plain* multiplication. It turns out that two other forms of multiplication are useful: the first is the *transposed* multiplication, closely related to the *middle product* of [9]. The other noteworthy form is the *short product*, introduced in [14] and studied in [10]. Both are detailed in Section 3, together with mentions of their applications.

Our contribution: code generation for divide-and-conquer algorithms. Beyond from the naive algorithm, the main classes of fast algorithms are generalizations of Karatsuba’s approach [11,18,21], where a given pattern is used in a divide-and-conquer fashion, and FFT-like approaches, that use or build suitable evaluation points [7,15,6], usually roots of unity.

In this paper, we focus on the former family. Despite its richness, we are not aware of a systematic treatment of algorithms for transposed or short product in this context. First, we fill this gap, giving explicit versions of such algorithms. We will see the algorithms of this family can be described in finite terms, by triples of graphs. Then, we describe a code generator that turns such graphs into C implementations, avoiding the need to reimplement everything from scratch; the performance of these implementations is among the best known to us.

Previous work. Most algorithms for plain multiplication discussed here are well-known: the most classical ones are due to Karatsuba [11] and Toom [18], with improvements in [1]; some less well-known ones are due to Winograd [21].

It is well-known that any algorithm for plain multiplication can be transformed into an algorithm for transposed multiplication; this is already in [21], and is developed in [9,3], together with applications. However, while the *existence* of transposed algorithms was known, our general derivation of *explicit* divide-and-conquer algorithms is new, to our knowledge: the only explicit examples in [9,3] describe Karatsuba multiplication. Similarly, the possibility of using any divide-and-conquer algorithm to perform short products is hinted at in [10], but no details are given; only the case of Karatsuba multiplication is developed in great detail. Our general presentation is, to our knowledge, new.

Computational model. Our problems are bilinear; computations will thus be done as follows: linear combinations of the inputs are computed (separately), followed by pairwise products of the values thus obtained; the result is deduced by a last series of linear combinations of these products. Our complexity estimates count the linear operations and the pairwise products.

2 Preliminaries: Graphs for Linear Maps

We address first the linear part of the computations: we recall here how to compute linear maps using a graphical representation. The material in this section is well-known [5, Ch. 13].

Definition. A *linear graph* \mathcal{G} consists of

- a directed acyclic graph (V, E) with k inputs and ℓ outputs,
- a weight function λ which assigns a weight $\lambda(e) \in \mathbb{R}$ to each edge e ,
- orderings (A_0, \dots, A_{k-1}) and $(F_0, \dots, F_{\ell-1})$ of the inputs and outputs.

One assigns a matrix to a linear graph in a straightforward way. Each vertex is assigned a value, obtained by following the “flow” from inputs to outputs: going from a vertex v to a vertex v' along an edge e , the value at v is multiplied by the weight $\lambda(e)$; the value at v' is obtained by summing the contributions of

all incoming edges. The values obtained at each vertex are linear combinations of the values a_0, \dots, a_{k-1} given at the inputs A_0, \dots, A_{k-1} . In particular, let $f_0, \dots, f_{\ell-1}$ be the values computed by the output nodes $F_0, \dots, F_{\ell-1}$; f_i can thus be written $f_i = L_{i,0}a_0 + \dots + L_{i,k-1}a_{k-1}$, for some constants $L_{i,j}$, so that

$$\begin{bmatrix} f_0 \\ \vdots \\ f_{\ell-1} \end{bmatrix} = \mathbf{L} \begin{bmatrix} a_0 \\ \vdots \\ a_{k-1} \end{bmatrix}, \quad \text{with } \mathbf{L} = \begin{bmatrix} L_{0,0} & \cdots & L_{0,k-1} \\ \vdots & & \vdots \\ L_{\ell-1,0} & \cdots & L_{\ell-1,k-1} \end{bmatrix}.$$

Thus, we say that the linear graph \mathcal{G} *computes* the matrix \mathbf{L} .

Cost. To measure the number of operations attached to a linear graph, we first make our computational model more precise: we count at unit cost multiplications by constants, as well as operations of the form $\alpha = \pm\beta \pm \gamma$. Then, we define the *cost* of \mathcal{G} as the number

$$c(\mathcal{G}) := |\{e \in E \mid \lambda(e) \neq \pm 1\}| + |E| - |V| + k.$$

We claim that, with $\mathbf{a} = [a_0 \ \cdots \ a_{k-1}]^t$, the matrix-vector product $\mathbf{a} \mapsto \mathbf{L}\mathbf{a}$ can be computed using $c(\mathcal{G})$ operations. Indeed, along the edges, each multiplication by a constant different from ± 1 costs one operation, which add up to $|\{e \in E \mid \lambda(e) \neq \pm 1\}|$. Then, if the input of a vertex v consists of s edges, computing the value at v uses another $s - 1$ operations of the form $\pm\beta \pm \gamma$; summing over all v gives an extra $|E| - |V| + k$ operations.

Transposition. The *transposition principle* asserts that an algorithm performing a matrix-vector product can be transposed, producing an algorithm that computes the transposed matrix-vector product, in almost the same complexity as the original one. In our model, the transposition principle is easy to prove. If \mathcal{G} is a linear graph with k inputs and ℓ outputs, that computes a matrix \mathbf{L} , we define the transposed graph \mathcal{G}^t exchanging inputs and outputs and reversing the edges, without changing the weights. Theorem 13.10 in [5] proves that \mathcal{G}^t computes the transposed matrix of \mathbf{L} ; besides the cost $c(\mathcal{G}^t)$ is given by $c(\mathcal{G}^t) = c(\mathcal{G}) - k + \ell$.

3 Polynomial Multiplication and Its Variants

In this section, we describe three variants of polynomial multiplication (plain, transposed and short product), and give algorithms for all of them. The algorithms we consider will be called “divide-and-conquer”, following the terminology of [19]. The most well-known representatives of this class are due to Karatsuba and Toom, though many more exist.

3.1 Divide-and-Conquer Algorithms

A *divide-and-conquer* algorithm of parameters (k, ℓ) , with $k < \ell$, is a triple $\mathcal{G} = (\mathcal{G}_A, \mathcal{G}_B, \mathcal{G}_C)$ of linear graphs such that \mathcal{G}_A and \mathcal{G}_B have k inputs and ℓ outputs, and \mathcal{G}_C has ℓ inputs and $2k - 1$ outputs (other conditions follow).

Let $\mathbf{A} = (A_0, \dots, A_{k-1})$ and $\mathbf{B} = (B_0, \dots, B_{k-1})$ be indeterminates and let $L_0(\mathbf{A}), \dots, L_{\ell-1}(\mathbf{A})$ and $M_0(\mathbf{B}), \dots, M_{\ell-1}(\mathbf{B})$ be the linear forms computed by respectively \mathcal{G}_A and \mathcal{G}_B . Let further $N_i = L_i M_i$ and let P_0, \dots, P_{2k-2} be the linear forms computed by \mathcal{G}_C . Then, the last conditions for \mathcal{G}_A , \mathcal{G}_B and \mathcal{G}_C to form a divide-and-conquer algorithm is that for $i = 0, \dots, 2k-2$,

$$P_i(\mathbf{N}) = \sum_{0 \leq j < k, 0 \leq j' < k, j+j'=i} A_j B_{j'},$$

where $P_i(\mathbf{N})$ stands for the evaluation of the linear form P_i at $N_0, \dots, N_{\ell-1}$. For instance, Karatsuba's algorithm has $k = 2, \ell = 3$ and

- $L_0 = A_0, L_1 = A_0 + A_1, L_2 = A_1$
- $M_0 = B_0, M_1 = B_0 + B_1, M_2 = B_1$
- $P_0(\mathbf{N}) = N_0, P_1(\mathbf{N}) = N_1 - N_0 - N_2, P_2(\mathbf{N}) = N_2$.

Other examples due to Toom [18] and Winograd [21] are in the last section; note that in these examples, $\mathcal{G}_A = \mathcal{G}_B$.

3.2 Plain Multiplication

Let $\mathcal{G} = (\mathcal{G}_A, \mathcal{G}_B, \mathcal{G}_C)$ be a divide-and-conquer algorithm of parameters (k, ℓ) . We now recall the well-known derivation of an algorithm for plain multiplication using \mathcal{G} ; note that this formalism does not cover evaluations at points in $\mathbb{R}(x)$, which are useful e.g. over $\text{GF}(2)$ [22].

Given n and A, B in $\mathbb{R}[x]_n$, we let $h = \lfloor (n+k-1)/k \rfloor$ and $h' = n - (k-1)h$, so that $h' \leq h$. To make the algorithm simpler, we also want $h' > 0$; this will be the case as soon as $n > (k-1)^2$. Then, we write

$$A = A_0 + A_1 x^h + \dots + A_{k-1} x^{(k-1)h}, \quad B = B_0 + B_1 x^h + \dots + B_{k-1} x^{(k-1)h},$$

$$C = C_0 + C_1 x^h + \dots + C_{2k-2} x^{(2k-2)h}.$$

In Algorithm [1](#) below, we use the notation $\text{slice}(A, p, q)$ to denote the ‘‘slice’’ of A of length q starting at index p , that is, $(A \text{ div } x^p) \bmod x^q$. Note that A_0, \dots, A_{k-2} are in $\mathbb{R}[x]_h$ and A_{k-1} in $\mathbb{R}[x]_{h'}$; the same holds for the B_i ; similarly, C_0, \dots, C_{2k-4} are in $\mathbb{R}[x]_{2h-1}$, C_{2k-3} in $\mathbb{R}[x]_{h+h'-1}$ and C_{2k-2} in $\mathbb{R}[x]_{2h'-1}$.

To obtain C , we compute the linear combinations L_i of A_0, \dots, A_{k-1} and M_i of B_0, \dots, B_{k-1} , the products $N_i = L_i M_i$, and the polynomials C_i as the linear combinations $P_i(\mathbf{N})$. To handle the recursive calls, we need bounds e_i and f_i such that $\deg(L_i) < e_i$ and $\deg(M_i) < f_i$ holds: we simply take $e_i = h$ if $L_i \neq A_{k-1}$, and $e_i = h'$ if $L_i = A_{k-1}$; the same construction holds for f_i . For simplicity, we assume that $e_i = f_i$ for all i : this is e.g. the case when $\mathcal{G}_A = \mathcal{G}_B$. If $e_i \neq f_i$, the recursive calls need to be slightly modified, by e.g. doing a recursive call in length $\min(e_i, f_i)$ and an extra $O(n)$ operations to complete the product.

The cost $T(n)$ of this algorithm is $O(n^{\log_k(\ell)})$; one cannot easily give a more precise statement, since the ratio $T(n)/n^{\log_k(\ell)}$ does not have a limit as $n \rightarrow \infty$. We give here closed form expressions for n of the form k^i ; in this case, we can go down the recursion until $n = 1$, which simplifies the estimates.

Algorithm 1. Mul(A, B, n)**Require:** A, B, n , with $\deg(A) < n$ and $\deg(B) < n$ **Ensure:** $C = AB$

```

1: if  $n \leq (k-1)^2$  then
2:   return  $AB$  naive multiplication
3:  $h = \lfloor (n+k-1)/k \rfloor$ ,  $h' = n - (k-1)h$ 
4: for  $i = 0$  to  $k-2$  do
5:    $A_i = \text{slice}(A, ih, h)$ 
6:    $B_i = \text{slice}(B, ih, h)$ 
7:  $A_{k-1} = \text{slice}(A, (k-1)h, h')$ 
8:  $B_{k-1} = \text{slice}(B, (k-1)h, h')$ 
9: compute the linear combinations  $L_0, \dots, L_{\ell-1}$  of  $A_0, \dots, A_{k-1}$ 
10: compute the linear combinations  $M_0, \dots, M_{\ell-1}$  of  $B_0, \dots, B_{k-1}$ 
11: for  $i = 0$  to  $\ell-1$  do
12:    $N_i = \text{Mul}(L_i, M_i, e_i)$ 
13: recover  $C_0, \dots, C_{2k-2}$  as linear combinations of  $N_0, \dots, N_{\ell-1}$ 
14: return  $C = C_0 + C_1x^h + \dots + C_{2k-2}x^{(2k-2)h}$ 

```

The number of bilinear multiplications is ℓ^i . As to the linear operations, let c_A, c_B, c_C be the costs of $\mathcal{G}_A, \mathcal{G}_B, \mathcal{G}_C$. On inputs of length n , a quick inspection shows that we do $c_A n/k + c_B n/k + c_C(2n/k - 1)$ operations at steps [9](#), [10](#) and [13](#) and $2(k-1)(n/k - 1)$ additions at step [14](#), for a total of $(c_A + c_B + 2c_C + 2k - 2)n/k - (c_C + 2k - 2)$. For $n = k^i$, summing over all recursive calls gives an overall estimate of

$$t(i) = (c_A + c_B + 2c_C + 2k - 2)(\ell^i - k^i)/(\ell - k) - (c_C + 2k - 2)(\ell^i - 1)/(\ell - 1). \quad (1)$$

3.3 Transposed Product

If A is fixed, the map $A, B \mapsto AB$ becomes linear in B . The *transposed product* is the transposed map; applications include Newton iteration [9](#), evaluation and interpolation [3](#), etc.

If A is in $\mathbb{R}[x]_n$, multiplication-by- A maps $B \in \mathbb{R}[x]_n$ to $C = AB \in \mathbb{R}[x]_{2n-1}$. For $k \in \mathbb{N}$, we identify $\mathbb{R}[x]_k$ with its dual; then, the transposed product $A, C \mapsto B = CA^t$ maps $C \in \mathbb{R}[x]_{2n-1}$ to $B \in \mathbb{R}[x]_n$. Writing down the matrix of this map, we deduce the explicit formula [93](#)

$$B = (C\tilde{A} \operatorname{div} x^{n-1}) \bmod x^n,$$

where $\tilde{A} = x^{n-1}A(1/x)$ is the reverse of A . This formula gives a quadratic algorithm for the transposed product; actually, any algorithm for the plain product can be used, by computing $C\tilde{A}$ and discarding the unnecessary terms.

However, one can do better. As a consequence of the transposition principle, algorithms for the plain product yield algorithms for the transposed one, with only $O(n)$ cost difference: this was mentioned in [21](#), and developed further in [9](#) and [3](#). However, none of the previous references gave an explicit form for the transposed version of divide-and-conquer algorithms, except for Karatsuba.

In Algorithm 2, we provide such an explicit form, on the basis of a divide-and-conquer algorithm \mathcal{G} of parameters (k, ℓ) . The polynomial A is subdivided as before, and the linear operations applied to the slices A_i are unchanged. The other input is now C ; we apply to it the transposes of the operations seen in Algorithm 1, in the reverse order. Summing C_0, \dots, C_{2k-2} in Algorithm 1 becomes here the subdivision of C into C_0, \dots, C_{2k-2} , using the degree information obtained in the previous section. Then, we follow the transposed graph \mathcal{G}_C^t to obtain $N_0, \dots, N_{\ell-1}$; we enforce the degree constraints $\deg(N_i) < 2e_i - 1$ by truncation (these truncations are the transposes of injections between some $\mathbb{R}[x]_{2e_i-1}$ and $\mathbb{R}[x]_{2e_j-1}$ that were implicit at step 13 of Algorithm 1). After this, we apply the algorithm recursively, follow the transposed graph \mathcal{G}_B^t to obtain B_0, \dots, B_{k-1} , and obtain B as the sum $B_0 + \dots + B_{k-1}x^{(k-1)h}$.

Algorithm 2. TranMul(A, C, n)

Require: A, C, n , with $\deg(A) < n$ and $\deg(C) < 2n - 1$

Ensure: $B = CA^t$

- 1: **if** $n \leq (k - 1)^2$ **then**
 - 2: **return** CA^t naive transposed multiplication
 - 3: $h = \lfloor (n + k - 1)/k \rfloor$, $h' = n - (k - 1)h$
 - 4: **for** $i = 0$ **to** $k - 2$ **do**
 - 5: $A_i = \text{slice}(A, ih, h)$
 - 6: $A_{k-1} = \text{slice}(A, (k - 1)h, h')$
 - 7: **for** $i = 0$ **to** $2k - 4$ **do**
 - 8: $C_i = \text{slice}(C, ih, 2h - 1)$
 - 9: $C_{2k-3} = \text{slice}(C, (2k - 3)h, h + h' - 1)$
 - 10: $C_{2k-2} = \text{slice}(C, (2k - 2)h, 2h' - 1)$
 - 11: compute the linear combinations $L_0, \dots, L_{\ell-1}$ of A_0, \dots, A_{k-1}
 - 12: compute the transposed linear combinations $N_0, \dots, N_{\ell-1}$ of C_0, \dots, C_{2k-2} , with N_i truncated modulo x^{2e_i-1}
 - 13: **for** $i = 0$ **to** $\ell - 1$ **do**
 - 14: $M_i = \text{TranMul}(L_i, N_i, e_i)$
 - 15: compute the transposed linear combinations B_0, \dots, B_{k-1} of $M_0, \dots, M_{\ell-1}$, with B_{k-1} truncated modulo $x^{h'}$ **output** $B_0 + \dots + B_{k-1}x^{(k-1)h}$
-

The cost $T'(n)$ of this algorithm is still $O(n^{\log_k(\ell)})$. Precisely, let c_A, c_B, c_C be the costs of $\mathcal{G}_A, \mathcal{G}_B, \mathcal{G}_C$, and consider the case where $n = k^i$. The number of bilinear multiplications does not change compared to the direct version. As to linear operations, the cost of step 12 is $(c_C - \ell + 2k - 1)(2n/k - 1)$ and that of step 15 is $(c_B - k + \ell)n/k$. After simplification and summation, we obtain that for $n = k^i$, the overall number of linear operations is now

$$t'(i) = (c_A + c_B + 2c_C + 3k - \ell - 2)(\ell^i - k^i)/(\ell - k) - (c_C + 2k - \ell - 1)(\ell^i - 1)/(\ell - 1).$$

With $t(i)$ given in Eq. (1), we obtain $t'(i) - t(i) = k^i - 1 = n - 1$, as implied by the transposition principle: the transposed algorithm uses $n - 1$ more operations.

3.4 Short Product

The *short product* is a truncated product: to A, B in $\mathbb{R}[x]_n$, it associates $C = AB \bmod x^n \in \mathbb{R}[x]_n$; it was introduced and described in [14, 10], and finds a natural role in many algorithms involving power series operations, such as those relying on Newton iteration [4]. The situation is similar to that of the transposed product: the previous references describe Karatsuba’s version in detail, but hardly mention other algorithms in the divide-and-conquer family. Thus, as for transposed product, we give here an explicit version of the short product algorithm, starting from a divide-and-conquer algorithm \mathcal{G} of parameters (k, ℓ) .

For Karatsuba’s algorithm, two strategies exist in the literature; the latter one, due to [10], extends directly to the general case. Instead of slicing the input polynomials, we “decimate” them: for $A \in \mathbb{R}[x]_n$, we write $A = \sum_{i < k} A_i(x^k)x^i$ (the same holds for B). Here, the polynomial A_i belongs to $\mathbb{R}[x]_{h_i}$, with $h_i = \lfloor (n + k - 1 - i)/k \rfloor$; we denote it by $A_i = \text{decimation}(A, i, h_i)$. Then, with $C_i = \sum_{j+j'=i} A_j B_{j'}$, we deduce

$$C = \sum_{i < 2k-1} C_i(x^k)x^i = \sum_{i < k-1} (C_i + xC_{i+k})(x^k)x^i + C_{k-1}(x^k)x^{k-1}.$$

We compute the linear combinations L_i of A_0, \dots, A_{k-1} and M_i of B_0, \dots, B_{k-1} , the products $N_i = L_i M_i$, and finally C_i using the linear forms P_0, \dots, P_{2k-2} . We need to compute C_i modulo x^{h_i} . For $i < \ell$, let thus i' be the largest index such that the product $L_i M_i$ appears with a non-zero coefficient in the linear form $P_{i'}$ (this depends on the divide-and-conquer algorithm), and let $g_i = h_{i'}$. Since the h_i form a decreasing sequence, it suffices to compute $L_i M_i \bmod x^{g_i}$.

These steps are summarized in Algorithm 3, where we reuse the notation introduced above. Here, it suffices that $n \geq k$ to ensure that all h_i , and thus all g_i , are positive, since smallest is $h_{k-1} = \lfloor n/k \rfloor$. As for the previous algorithms, the cost is $O(n^{\log_k(\ell)})$; however, the precise analysis is much more delicate [10], so we do not give any closed-form estimate here, even for n of the form k^i .

Algorithm 3. ShortMul(A, B, n)

Require: A, B, n , with $\deg(A) < n$, $\deg(B) < n$

Ensure: $C = AB \bmod x^n$

- 1: **if** $n = 1$ **then**
 - 2: **return** AB
 - 3: **for** $i = 0$ **to** $k - 1$ **do**
 - 4: $A_i = \text{decimation}(A, i, h_i)$
 - 5: $B_i = \text{decimation}(B, i, h_i)$
 - 6: compute the linear combinations $L_0, \dots, L_{\ell-1}$ of A_0, \dots, A_{k-1}
 - 7: compute the linear combinations $M_0, \dots, M_{\ell-1}$ of B_0, \dots, B_{k-1}
 - 8: **for** $i = 0$ **to** $\ell - 1$ **do**
 - 9: $N_i = \text{ShortMul}(L_i \bmod x^{g_i}, M_i \bmod x^{g_i}, g_i)$
 - 10: compute the linear combinations C_0, \dots, C_{2k-2} of $N_0, \dots, N_{\ell-1}$, truncating C_i modulo x^{h_i}
 - 11: **return** $C = \sum_{i=0}^{k-2} (C_i + xC_{i+k})(x^k)x^i + C_{k-1}(x^k)x^{k-1}$
-

For a given multiplication type (plain, transposed or short) and a given data type, we produce several functions: a top-level function, which allocates some workspace and does some precomputations (e.g., the modular inverses or roots of unity needed for the linear combinations), the main recursive function, and functions for the linear combinations.

Memory management. Intermediate results are stored in temporary memory, in successive slots of length either $\lceil n/k \rceil$ or $2\lceil n/k \rceil$. At code generation, when we determine that a memory area can be reused, we reuse it. We can thus determine how much workspace will be needed in a single call to the main recursive function in length n . In general, if the call in length n uses $rn + s$ space, the *total* amount will be $rn + rn/k + \dots + s \log_k(n) \leq rkn/(k-1) + s \log_k(n)$. This memory is allocated by the top-level function. Efforts are made to avoid using too much memory, similarly to what one would do when writing the code by hand. When an output of the linear combinations aliases an input, we reuse the input in all other operations (e.g., for Karatsuba, the linear combinations are $L_0 = a_0, L_1 = a_0 + a_1, L_2 = a_1$: no copy is made and only an addition takes place).

Naive product. We implemented naive algorithms (for plain, transposed and short products), for degrees up to 16. Our code for this case is generated automatically as well, so as to unroll loops, since the compiler was not doing a very good job by itself. We do not perform modular reduction after each step: we first compute the whole result without any reduction, and apply the reduction in the end. In degree $< n$, this reduces the number of reductions from n^2 to n . However, this slightly reduces the possible size of the modulus: only 60-bit modulus can now be used. No assembly code was used: using gcc's custom `__uint128_t` type, we obtained code of satisfying quality after compilation.

5 Experiments

We finally give the results of experiments on an Intel Core2 Duo CPU T7300 with 4Gb RAM, set to 800Mhz clock speed. The timings are in seconds, for 500 repetitions of the same computation. Our experiments use Karatsuba's algorithm and its generalizations by Toom, of parameters $(k, 2k-1)$: for the standard evaluation points $(0, \pm 1, \pm 2, \pm 1/2, \dots, \infty)$, we use linear graphs from [1]. Computing modulo p , with $p = 4r+1$, we wrote a version of Toom's algorithm (called i-Toom below) that evaluates at $(0, \pm 1, \pm \sqrt{-1}, 2, \infty)$ using FFT techniques in size 4. We also use a less known algorithm of parameters $(3, 6)$ due to Winograd, with only additions and subtractions in its linear combinations [21, Ch. IVc]. Complexity predicts that it should be slower than Karatsuba, but the simple structure of the linear combinations made it worthwhile to experiment with.

Comparison between divide-and-conquer algorithms. Figure 1 compares the algorithms of Karatsuba, Toom ($k=3$) and Winograd, for plain product, using `unsigned longs` (transposed and short products behave similarly). As predicted, Winograd's algorithm does not perform very well. More surprisingly,

Toom’s algorithm appears useful for most degrees (examples using other divide-and-conquer algorithm are given below). Jumps appear for all algorithms; these are due to crossing degree thresholds determined by both the parameter k of the graphs, and the threshold for the switch to the naive algorithm; increasing the latter smooths the curves noticeably. Finally, profiling using Valgrind shows that in all cases, 65% to 70% is the time is spent in the naive algorithm.

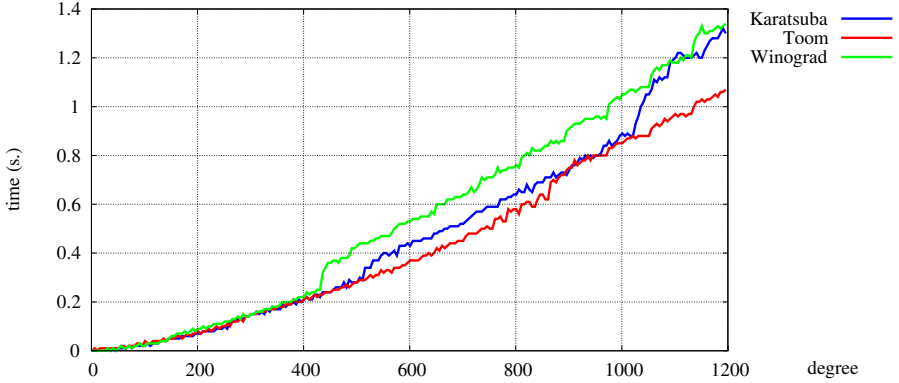


Fig. 1. Comparison between divide-and-conquer algorithms

Comparison between multiplication types. Figure 2 compares plain, transposed and short product, square and short square, with unsigned longs and Toom’s algorithm ($k = 3$); the results for other divide-and-conquer algorithms are similar. The transposed product is faster than its plain counterpart, even if operation count predicts it should be slightly slower. Indeed, in the naive transposed product, fewer modular reduction are needed than in the plain one (since the output is twice as short); this is not accounted for in our model and seems to explain the savings. The time for a short product is about 60% to 70% that of a plain product, as in 10 for Karatsuba. The square product and the short square are faster than their non-square counterparts, but not by much.

Comparison with other systems. For primes of size 60 bit, the library NTL v5.5 16 and the computer algebra system Magma v2.15-6 2 are the fastest implementations known to us. We use the two available representations for NTL, `lzz_p` and `ZZ_p` (our 60 bit primes are too large for the former, so we used 52 bit primes in that case). Figure 3 gives running times, where our code uses “standard” Toom multiplication for $k = 3$ or $k = 4$ or i-Toom for $p = 4r + 1$. Even though some other implementations use asymptotically faster algorithms (the staircases indicate FFT multiplication), our code performs better in these degree ranges. From degree 10000 on, Toom’s algorithm with $k = 5$ is the best of our divide-and-conquer algorithms, but does no better than NTL’s FFT.

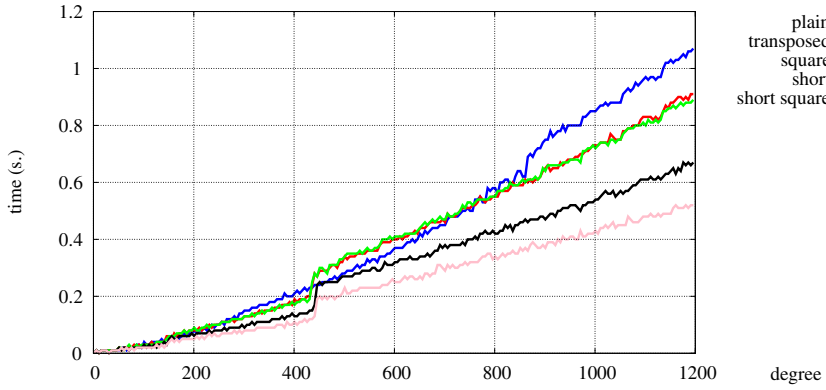


Fig. 2. Comparison between multiplication types

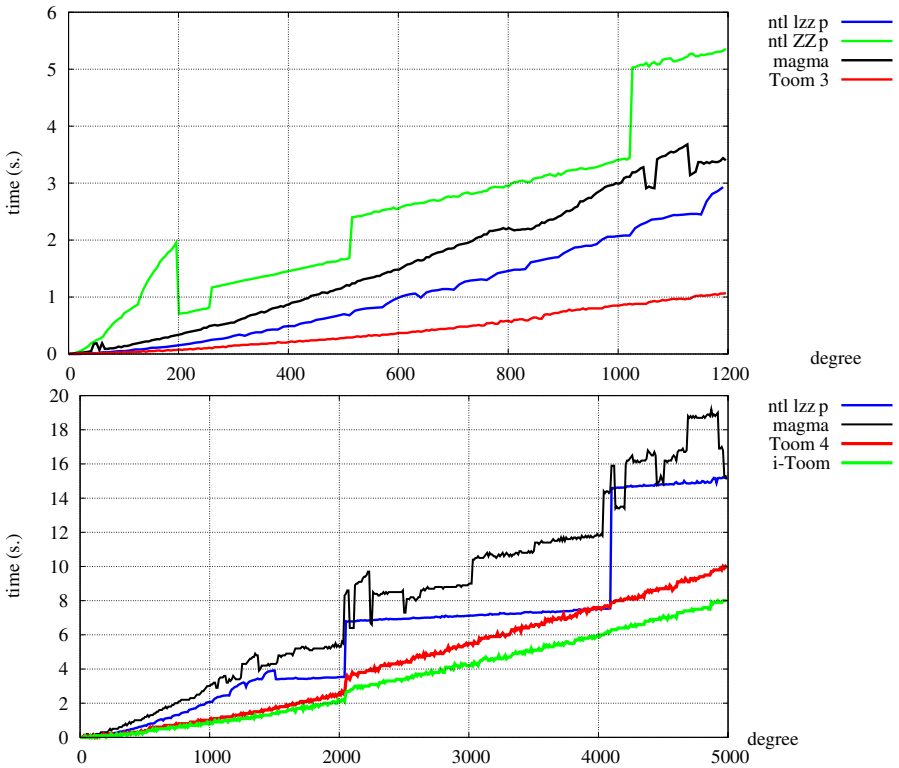


Fig. 3. Comparison with other systems

Comparison between data types. Operations with `double` coefficients are faster than with `unsigned longs`, but only by a factor of about 1.7 to 1.9 (for all variants, and for all divide-and-conquer algorithms). Divide-and-conquer algorithms do poorly in terms of precision with `double` coefficients: for useful kinds of inputs (such as solutions of ODE's), the cancellation errors make results unusable for degrees from 50 on.

6 Conclusion

Our approach offers several advantages: after paying the small price of writing the code generator, it becomes straightforward to experiment various divide-and-conquer algorithms, test optimizations, etc. Also, we now have general versions of transposed and short product. For plain products, performance is comparable to, and actually better than, that of software using FFT multiplication in significant degree ranges. For short products, our advantage is actually higher, since it is rather difficult to obtain an efficient short product using FFT multiplication.

Acknowledgments. We acknowledge the support of the Canada Research Chairs Program, of the MITACS MOCAA project and of NSERC, and thank the referees for their helpful comments.

References

1. Bodrato, M., Zanoni, A.: Integer and polynomial multiplication: towards optimal Toom-Cook matrices. In: ISSAC 2007, pp. 17–24. ACM, New York (2007)
2. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. *J. Symbolic Comput.* 24(3-4), 235–265 (1997)
3. Bostan, A., Lecerf, G., Schost, É.: Tellegen's principle into practice. In: ISSAC 2003, pp. 37–44. ACM, New York (2003)
4. Brent, R.P., Kung, H.T.: Fast algorithms for manipulating formal power series. *J. ACM* 25(4), 581–595 (1978)
5. Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic complexity theory. *Grund. Math. Wissen*, vol. 315. Springer, Heidelberg (1997)
6. Cantor, D.G., Kaltofen, E.: On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica* 28(7), 693–701 (1991)
7. Cooley, J., Tukey, J.: An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation* 19 (1965)
8. Gaudry, P., Thomé, E.: The mpFq library and implementing curve-based key exchanges. In: SPEED, pp. 49–64 (2007)
9. Hanrot, G., Quercia, M., Zimmermann, P.: The middle product algorithm. I. *Appl. Algebra Engrg. Comm. Comput.* 14(6), 415–438 (2004)
10. Hanrot, G., Zimmermann, P.: A long note on Mulders' short product. *J. Symb. Comput.* 37(3), 391–401 (2004)
11. Karatsuba, A., Ofman, Y.: Multiplication of multidigit numbers on automata. *Soviet Math. Dokl.* 7, 595–596 (1963)

12. Monagan, M., Pearce, R.: Parallel sparse polynomial multiplication using heaps. In: ISSAC 2009. ACM, New York (to appear, 2009)
13. Montgomery, P.L.: Modular multiplication without trial division. *Mathematics of Computation* 44(170), 519–521 (1985)
14. Mulders, T.: On short multiplications and divisions. *Appl. Algebra Engrg. Comm. Comput.* 11(1), 69–88 (2000)
15. Schönhage, A., Strassen, V.: Schnelle Multiplikation großer Zahlen. *Computing* 7, 281–292 (1971)
16. Shoup, V.: A library for doing number theory, <http://www.shoup.net/ntl/>
17. Shoup, V.: A new polynomial factorization algorithm and its implementation. *J. Symb. Comp.* 20(4), 363–397 (1995)
18. Toom, A.: The complexity of a scheme of functional elements realizing the multiplication of integers. *Doklady Akad. Nauk USSR* 150(3), 496–498 (1963)
19. van der Hoeven, J.: Relax, but don't be too lazy. *J. Symbolic Comput.* 34(6), 479–542 (2002)
20. von zur Gathen, J., Gerhard, J.: *Modern computer algebra*, 2nd edn. Cambridge University Press, Cambridge (2003)
21. Winograd, S.: Arithmetic complexity of computations. In: *CBMS-NSF Regional Conference Series in Applied Mathematics*, vol. 33. SIAM, Philadelphia (1980)
22. Zimmermann, P.: Irred-ntl patch, <http://www.loria.fr/~zimmerma/irred/>

Solving Structured Polynomial Systems and Applications to Cryptology (Plenary Talk)

Jean-Charles Faugère

SALSA Project INRIA, Centre Paris-Rocquencourt
UPMC, Univ Paris 06, LIP6
CNRS, UMR 7606, LIP6
UFR Ingénierie 919, LIP6 Passy Kennedy
Boite Courrier 169, 4, Place Jussieu 75252 Paris Cedex 05
Jean-Charles.Faugere@inria.fr
<http://www-salsa.lip6.fr/~jcf>

Algebraic Cryptanalysis

Cryptography is a collection of mathematical techniques used to secure the transmission and storage of information. A *fundamental* problem in cryptography is to *evaluate the security of cryptosystems* against the most powerful techniques. To this end, several *general* methods have been proposed: linear cryptanalysis, differential cryptanalysis, ... Extensively used cryptographic standards – such as AES [1] – are all resistant against linear and differential attacks. In this talk, we will describe another general method – *Algebraic Cryptanalysis* – which can be used to evaluate the security of such cryptosystems.

Algebraic cryptanalysis can be described as a general framework that permits to evaluate the security of a wide range of cryptographic schemes. The basic principle of such cryptanalysis is to model a cryptographic primitive by a set of multivariate polynomial equations. The system of equations is constructed in such a way that solving the system is equivalent to recover a secret information of the cryptographic primitive (for instance, the secret key in the case of an encryption scheme). Consequently, evaluate the security of this cryptosystem is equivalent to estimate the theoretical and practical complexity of solving the corresponding system of equations. Since one of the most efficient tool for solving algebraic system over finite field is Gröbner bases [2], it is necessary to evaluate theoretically (e.g. [3]) and practically (e.g. [8]) the complexity of computing Gröbner bases over \mathbb{F}_q .

While it is well known that solving system of polynomial equations is NP-hard [4] in many applications, including cryptography, the polynomial systems that we have to consider are *not random* at all (see for instance [6]). Hence, it is a crucial task to identify several classes of polynomial systems that are easier to solve (or at least such that we are able to predict accurately the complexity [5]). In this

talk we will consider a public-key cryptosystem (namely the Minrank problem) and we will show [7] how its multi-homogenous structure can be used to predict accurately the complexity of the Gröbner basis computation. For instance, for a recommended family of parameters, we can solve the corresponding systems in polynomial time and thus break the corresponding cryptosystem.

References

1. Daemen, J., Rijmen, V.: *The Design of Rijndael: The Wide Trail Strategy*. Springer, Heidelberg (2001)
2. Buchberger, B.: *An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal (German)*, PhD Thesis, Univ of Innsbruck, Math. Institute, Austria, English Translation: *J. of Symbolic Computation*, Special Issue on Logic, Math and Comp Science: *Interactions* 41(3-4), 475-511 (1965)
3. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: *Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems*. In: *Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry* (2005)
4. Garey, M.R., Johnson, D.B.: *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York (1979)
5. Courtois, N.: *Efficient Zero-knowledge Authentication Based on a Linear Algebra Problem MinRank*. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, p. 402. Springer, Heidelberg (2001)
6. Faugère, J.-C., Joux, A.: *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems using Gröbner bases*. In: Boneh, D. (ed.) *CRYPTO 2003*. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)
7. Faugère, J.-C., Levy-dit-Vehel, F., Perret, L.: *Cryptanalysis of MinRank*. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 280–296. Springer, Heidelberg (2008)
8. Faugère, J.-C.: *A New Efficient Algorithm for Computing Gröbner Basis without Reduction to Zero: F_5* . In: *Proceedings of ISSAC, July 2002*, pp. 75–83. ACM press, New York (2002)

The Comparison Method of Physical Quantity Dimensionalities

Alexander V. Flegontov¹ and M.J. Marusina²

¹ Herzen State Pedagogical University of Russia,
48, Moika Emb., 191186, St.-Petersburg, Russia
aflegontoff@herzen.spb.ru

² St.Petersburg State University of Information Technology, Mechanics and Optics

Abstract. There is examined the comparison method of dimensionalities on a basis of group-theoretical analysis. It has obtained certain scaling group and operators possible within the differential equations for different tasks. It has made the comparison of calculated operators of tension with known results.

1 Introduction

Methods of similarity and dimensionality theory have huge practical and theoretical value [1 – 3]. It is possible to argue on a certain analogy of dimensionality theory (similarity theory) and the fundamental theory for contemporary mathematics and physics – geometrical theory of invariants concerning the transformation of coordinates.

For the correct formulation and processing of experiments, which results would allow determine general trends, it is essentially important to select non-dimensional parameters. Their amount has to be minimal, and selected parameters should reflect the main characteristics of physical process in the most convenient form. The probability of such preliminary qualitative-theoretical analysis and the choice of the defining non-dimensional parameters system derive from the dimensionality and similarity theory [1].

Also it is necessary to mention the opposite influence of dimensionality theory on group-theoretical analysis tasks, which is entirely used in different applications. Such method is known as the dimensionalities comparison [3].

2 The Theory of Dimensionalities

The notion of scaling group is closely connected with the theory of physical quantity dimensionalities [3]. Let us consider the main equation of dimension

$$X = \langle X \rangle [X], \quad (1)$$

which represents the measurable physical quantity X as a product of its numerical value $\langle X \rangle$ and the unit of physical quantity $[X]$.

If through E_α , $\alpha = 1, \dots, r$, we denote some independent units of physical quantity, and if it is possible to express E in terms of these units, and such an expression would have the form of compound monomial: $E = E_1^{\lambda_1} \dots E_r^{\lambda_r} = \dim X$, then it could be called the dimensionality of physical quantity X in the units E_α . Hence, the following form is valid for the physical quantity X :

$$X = \langle X \rangle \dim X = \langle X \rangle E_1^{\lambda_1} \dots E_r^{\lambda_r}. \quad (2)$$

It follows from this formula that multiplication and involution (in any real power) of physical quantities are reduced to the same operation on their numerical values and dimensionalities. Further, if the physical quantity X is the function of physical quantity x , then dimensionality of derivative would be determined according to the rule $\dim \partial_x X = \dim X / \dim x$.

The quantity X with zero dimension ($\lambda_1 = \dots = \lambda_r = 0$) is called non-dimensional. In other words, non-dimensional quantity is such physical quantity, in which dimensionality of the main physical quantities are included in zero power. The dimensionality is not a kind of invariable property of this physical quantity and depends on the way of constructing the system of units. For example, in international system of units SI (*LMTIQNJ*), where these symbols are the symbols of basic quantities: length L , mass M , time T , electric current strength I , thermodynamic temperature Q , matter quantity N and light intensity J , the permittivity (ratio of electric flux density and electric field intensity) z has the following dimensionality $\dim z = L^{-3}M^{-1}T^4I^2$. In the electrostatic system SGSE (*LMT*), where basic units are centimeter, gram, second, the permittivity z is a dimensionless quantity and for the vacuum is equal to unity $z_0 = 1$ (z_0 is the electric constant).

According to the work [3], quantities X_1, \dots, X_N will call dependent (on their dimensionalities), if there exist such numbers χ_1, \dots, χ_N , not all simultaneously equal to zero, with which the combination of these quantities $X_1^{\chi_1} \dots X_N^{\chi_N}$ is non-dimensional. Otherwise quantities X_1, \dots, X_N would be called independent.

One of the essential tasks of dimensionality theory consists in determining how many independent physical quantities there are among physical quantities of such gang X_1, \dots, X_N choosing them and expressing the rest quantities by independents.

Example 1. In mechanical systems, the following basic quantities are considered as independent units: $E_1 = L = m$, $E_2 = M = kg$, $E_3 = T = s$. Hence, dimensions of velocity v , acceleration g , substance density r , and pressure p could be written in such a form:

$$\begin{aligned} \dim v = ms^{-1} = LT^{-1} = E_1 E_3^{-1}, \quad \dim g = ms^{-2} = LT^{-2} = E_1 E_3^{-2}, \\ \dim r = kgm^{-3} = ML^{-3} = E_1^{-3} E_2, \\ \dim p = kgm^{-1}s^{-2} = ML^{-1}T^{-2} = E_1^{-1} E_2 E_3^{-2}. \end{aligned}$$

Here v , g , r are independent quantities since there are no numbers χ_1, \dots, χ_N that are not simultaneously equal to zero and with which the combination of

these quantities v , g , r would be non-dimensional. Let us show this examining the combination $v^{\chi_1} g^{\chi_2} r^{\chi_3}$ or

$$(E_1 E_3^{-1})^{\chi_1} (E_1 E_3^{-2})^{\chi_2} (E_1^{-3} E_2)^{\chi_3} = E_1^{\chi_1 + \chi_2 - 3\chi_3} E_2^{\chi_3} E_3^{-\chi_1 - 2\chi_2}.$$

Equating to zero exponents E_1 , E_2 , E_3 , we will obtain the system of equations. While solving it we will calculate: $\chi_1 = \chi_2 = \chi_3 = 0$. Hence, all exponents are simultaneously equal to zero, in such a case there is no dimensionless combination of quantities v , g , r , and these quantities are independent in accordance with the above definition.

Let us consider the combination of quantities v , r , p : $v^{\chi_1} r^{\chi_2} p^{\chi_3}$,

$$(E_1 E_3^{-1})^{\chi_1} (E_1^{-3} E_2)^{\chi_2} (E_1^{-1} E_2 E_3^{-2})^{\chi_3} = E_1^{\chi_1 - 3\chi_2 - \chi_3} E_2^{\chi_2 + \chi_3} E_3^{-\chi_1 - 2\chi_3},$$

solving the system, we will obtain the following result: χ_1 is any number, $\chi_2 = \frac{1}{2}\chi_1$, $\chi_3 = -\frac{1}{2}\chi_1$. It means that there is such a set of exponents χ_1 , χ_2 , χ_3 , under which not all of them are simultaneously equal to zero, and with such a set the combination of quantities v , r , p will be dimensionless. For example, if $\chi_1 = 1$ then

$$\chi_2 = \frac{1}{2}, \quad \chi_3 = -\frac{1}{2}, \quad \dim v \left(\frac{r}{p}\right)^{\frac{1}{2}} = E_1 E_3^{-1} \left(\frac{E_1^{-3} E_2}{E_1^{-1} E_2 E_3^{-2}}\right)^{\frac{1}{2}} = 1.$$

Hence, since there exists a non-dimensional combination of quantities v , r , p , so velocity, density, and pressure are dependent variables.

Let us consider the transition from one set of independent basic units E_α , $\alpha = 1, \dots, r$ to the other set. The change of dimension scale by transition from units E_α to new units E'_α , i.e.,

$$E_\alpha = a^\alpha E'_\alpha, \quad \alpha = 1, \dots, r \quad (3)$$

will lead to a change in the numerical value of quantity X in accordance with the equality following from formula (2)

$$\langle X \rangle \dim X = \langle X \rangle (a^1)^{\lambda_1} \dots (a^r)^{\lambda_r} (E'_1)^{\lambda_1} \dots (E'_r)^{\lambda_r} = \langle X \rangle' \dim X',$$

where $\dim X' = (E'_1)^{\lambda_1} \dots (E'_r)^{\lambda_r}$ – is the dimensionality of X in new units, and

$$\langle X \rangle' = \langle X \rangle \prod_{\alpha=1}^r (a^\alpha)^{\lambda_\alpha} \quad (4)$$

The analysis of equality (4) demonstrates that any change of the numerical value of quantity X during the passage to new dimensional units is just a group transformation belonging to scaling group H^r [3]. Hence it follows the assertion: the quantity X is non-dimensional if and only if its numerical value $\langle X \rangle$ is an invariant of the relevant scaling group H^r . That's why any result of the dimensionality theory could be obtained as some result of the scaling group theory.

The general conclusion of the dimensionality theory is well-known as the *Pi* - theorem: any non-dimensional function of physical quantities is the function of a non-dimensional combination of these quantities. Any parity between physical quantities is equivalent to certain parity between their non-dimensional combinations [3].

There is the opposite influence of dimensionality theory on group-theoretical analysis tasks. Mathematical equality expressing the connection between different physical quantities ought to possess the dimensional homogeneity. For differential equations, an admitted scaling group could be calculated by introducing few independent units and searching dimensionalities of the rest of quantities by using equations themselves on a basis of the request of dimensional equality of separate summands in these equations. Further, applying this method of comparison of dimensionalities, there have been obtained operators of infinitesimal symmetry in Lie algebra, generated by the scaling group, which are admitted by differential equations.

3 Calculation of Admitted Lie Operators of Scaling Group

3.1 The Heat Equation

Lets examine heat equation for one-dimensional rod

$$u_t = u_{xx}. \quad (5)$$

We assume the diffusion coefficient to be equal to unity. The reduced notations with relevant indexes are used for partial derivatives, for example, $u_t = \partial u / \partial t$, $u_{xx} = \partial^2 u / \partial x^2$, and so on. In equation (5), there exist three defining quantities: u , x , t , hence, according to the work [5], the number of independent quantities should be not less than two.

Example 2. As independent dimensional units we can choose

$$\dim u = E_1, \dim t = E_2,$$

so the dimensionality of x will be $\dim x = E_1^{\alpha_1} E_2^{\alpha_2}$. Let us write down the dimensionality equation for heat equation (5)

$$E_1 E_2^{-1} = E_1^{1-2\alpha_1} E_2^{-2\alpha_2}.$$

The comparison of dimensionalities leads to the system of linear equations for exponents, which has the unique solution. As a result, we have obtained the following table of exponents determining dimensionalities of all variables involved in equation (5):

	u	x	t
E_1	1	0	0
E_2	0	$\frac{1}{2}$	1

Under this table, it is possible to write down at once the operators of Lie algebra of infinitesimal symmetry generated by scaling group admitted by equation (5):

$$u\partial_u, \quad x\partial_x + 2t\partial_t.$$

Example 3. If we choose as independent dimension the units

$$\dim u = E_1, \dim x = E_2,$$

then $\dim t = E_1^{\alpha_1} E_2^{\alpha_2}$, and the equation of dimensionalities for (5) would obtain the form $E_1^{1-\alpha_1} E_2^{-\alpha_2} = E_1 E_2^{-2}$. Thus, the table of exponents determining dimensionalities of all variables involved in equation (5) will revert to the following form:

	u	x	t
E_1	1	0	0
E_2	0	1	2

In this case, the operators of Lie algebra of scaling group admitted by equation (5) will be identical to those calculated in the first example:

$$u\partial_u, \quad x\partial_x + 2t\partial_t.$$

So the operators of scaling group are included in the symmetry algebra of the heat equation, hence, this equation is invariant under the transformations of scaling group.

In the work of P. Olver [4, p.165] for the scaling group there exist the following operators of Lie algebra calculated by the traditional classic method through defining equations of the general symmetry group of the heat equation:

$$v_3 = u\partial_u, \quad v_4 = x\partial_x + 2t\partial_t.$$

Hence, it is possible to conclude that results calculated by the dimensionalities comparison method are equal to calculation made by P. Olver.

3.2 The KdV Equation

As example of equation of the high order we can examine the KdV equation

$$u_t + u_{xxx} + uu_x = 0. \tag{6}$$

This equation is typical of the shallow water theory and of other physical systems concerned with non-linear effects and dispersion.

In equation (6), there are three defining quantities: u , x , t , and so the number of independent quantities ought to be not less than one. Let us take as an independent dimensional unit u :

$$\dim u = E,$$

so dimensionalities x and t : $\dim x = E^\alpha$, $\dim t = E^\beta$. Hence, the table of exponents determining dimensionalities of all variables involved in equation (6) will have the form

	u	x	t
E	1	$-\frac{1}{2}$	$-\frac{3}{2}$

Operators of Lie algebra of scaling group admitted by the equation (6):

$$x\partial_x + 3t\partial_t - 2u\partial_u.$$

If we select x or t as independent dimensional units, operators of scaling group would be the same.

Thus, for the KdV equation, the infinitesimal symmetries for scaling group calculated by the dimensionalities comparison method are completely equal to calculations of P. Olver. In the work [4, p. 174] it is resulted the symmetry algebra of the KdV equation generated by four vector fields, particularly for scaling

$$v_4 = x\partial_x + 3t\partial_t - 2u\partial_u.$$

3.3 The Boundary Layer Equation

Let us examine the equations of two-dimensional non-stationary boundary layer (the Prandtl equations) that are well-known from the hydrodynamics, which describe the motion of ductile incompressible liquid nearby the impenetrable solid surface along with the enormous Reynolds numbers. Researched system of equations according to the work [3] has the following form:

$$\{u_t + uu_x + vu_y + p_x = u_{yy}, p_y = 0, u_x + v_y = 0. \tag{7}$$

Here u and v are coordinates of the velocity vector; x, y are space coordinates; t is the time; p is the pressure, besides, in this system of equations, the solidity of liquid and ductility coefficient are equal to unity.

In system (7), there are six defining quantities: u, v, x, y, t, p , so the number of independent quantities ought to be not less than two. As independent dimensional units we can choose $\dim t = E_1$ and $\dim p = E_2$. If we make an assumption that the dimensionalities of the rest of variables are possible to write in a such way

$$\dim x = E_1^{\alpha_1} E_2^{\alpha_2}, \dim y = E_1^{\beta_1} E_2^{\beta_2}, \dim u = E_1^{\gamma_1} E_2^{\gamma_2}, \dim v = E_1^{\delta_1} E_2^{\delta_2}$$

then the dimensionality equation for system (7) would have the following form:

$$E_1^{\gamma_1-1} E_2^{\gamma_2} + E_1^{2\gamma_1-\alpha_1} E_2^{2\gamma_2-\alpha_2} + E_1^{\delta_1+\gamma_1-\beta_1} E_2^{\delta_2+\gamma_2-\beta_2} + E_1^{-\alpha_1} E_2^{1-\alpha_2} = E_1^{\gamma_1-2\beta_1} E_2^{\gamma_2-2\beta_2}.$$

As a result of dimensionalities comparison, it has been formulated the table of exponents determining the dimensionalities of all variables encompassed by system (7):

	t	x	y	u	v	p
E_1	1	1	$\frac{1}{2}$	0	$-\frac{1}{2}$	0
E_2	0	$\frac{1}{2}$	0	$\frac{1}{2}$	0	1

Hence, operators of scaling group in the Lie algebra admitted by the system of equations of the two-dimensional non-stationary boundary layer (7) are as follows:

$$x\partial_x + u\partial_u + 2p\partial_p, \quad -2t\partial_t - 2x\partial_x - y\partial_y + v\partial_v.$$

These operators coincide with scaling operators obtained in the work [3, p. 131] by means of resolving defining equations for the system of equations of boundary layer. Besides, these operators coincide with scaling operators obtained in the work [3] by the dimensionality theory. However, u and v were selected as independent quantities, which actually are dependent because their combination is dimensionless.

The symmetry algebra of examined equations contains certain operators of scaling group. Hence, these equations are invariant under the transformations of scaling group.

3.4 The Equation of Self-balancing Beam Oscillation

It is known that if the beam is under the harmonic external influence $F_\sim = F_m \sin \omega t$, where F_m and ω are the amplitude and circular frequency of external force, correspondingly, so its dynamics is described by the differential equation

$$T_K^2 \frac{\partial^2 \varphi}{\partial t^2} + 2\xi T_K \frac{d\varphi}{dt} + \varphi = S_K F_m \sin \omega t, \quad (8)$$

where T_K is the time constant of beam; ξ is the power of tranquility; S_k is static coefficient of beams conversion (sensitivity). Dimensionalities of defining quantities involved in equation (8): $\dim T_K = T^1$, $\dim \varphi_0 = 1$, $\dim t = T$, $\dim \xi = 1$, $\dim S_K = M^{-1}T^{-2}$, $\dim F_m = MT^2$, $\dim \omega = T^{-1}$.

If as independent quantities we select $\dim T_K = E_1$, $\dim S_K = E_2$, then the rest of defining quantities could be written as $\dim t = E_1^{\alpha_1} E_2^{\alpha_2}$, $\dim F_m = E_1^{\beta_1} E_2^{\beta_2}$. Using these dimensionalities in equation (8), we will obtain the expression:

$$E_1^{2-2\alpha_1} E_2^{-2\alpha_2} = E_1^{\beta_1} E_2^{1+\beta_2} - E_1^{1-\alpha_1} E_2^{-\alpha_2}$$

in which let us unite the dimensionalities exponents in the following table:

	T_K	t	S_K	F_m
E_1	1	1	0	0
E_2	0	0	1	-1

In this case, the operators of scaling group of Lie algebra admitted by the equation of beams dynamics (8) having two independent quantities will have the following form:

$$T_K \partial_{T_K} + t \partial_t, \quad S_K \partial_{S_K} - F_m \partial_{F_m}. \quad (9)$$

According to the work [5], the operators of scaling group of Lie algebra obtained by the suggested method are equal for all four possible combinations of independent quantities:

$$\dim T_K = E_1, \dim S_K = E_2; \quad \dim S_K = E_1, \dim t = E_2;$$

$$\dim T_K = E_1, \dim F_m = E_2; \quad \dim F_m = E_1, \dim t = E_2;$$

Hence, the differential equation of the beams dynamics in self-balancing beam-balances (8) admits the scaling group with operators (9) or scaling group is the group symmetry of equation (8).

4 Conclusion

So, the dimensionalities comparison method allows us to find rather simply the operators of scaling group admitted by differential equations, and so, it allows arguing about invariance of equations under the transformations of scaling group. This method is certainly an efficient computing procedure of finding infinitesimal symmetries generated by transformations of scaling group.

References

1. Birkhoff, G.: Dimensional analysis of partial differential equations. *J. Electr. Engng.* 67, 1185–1188 (1948)
2. Buckingham, E.: Model experiments and the forms of empirical equations. *Trans. Amer. Soc. Mech. Eng.* 37, 263–296 (1915)
3. Ovsyannikov, L.V.: *Group Analysis of Differential Equations*. Nauka, Moscow (1978) (in Russian)
4. Olver, P.J.: *Applications of Lie Groups to Differential Equations*. Springer, New York (1986)
5. Marusina, M.J., Flegontov, A.V.: Applications in mechanics of the dimensionalities theory and groups theory. *Scientific Instrument Making* 15(1), 94–99 (2005)

Ambient Isotopic Meshing for Implicit Algebraic Surfaces with Singularities (Plenary Talk)

Jin-San Cheng, Xiao-Shan Gao, and Jia Li

Key Laboratory of Mathematics Mechanization
Institute of Systems Science, AMSS, Chinese Academy of Sciences
xgao@mmrc.iss.ac.cn

Abstract. A complete method is proposed to compute an ambient isotopic meshing for an implicit algebraic surface with singularities. By ambient isotopic, we mean a meshing with correct topology and any given precision. We use symbolic computation to guarantee the correctness and use numerical computation whenever possible to enhance the efficiency. Nontrivial examples are given to show the effectiveness of the algorithm.

1 The Main Results

To determine the topology of an algebraic surface and to use triangular meshes to approximately represent the surface are basic operations in computer graphics and geometric modeling. A recent survey on this topic can be found in [2].

A meshing is called **isotopic** if it has the same topology and the same geometry as the surface. A meshing is called **ambient isotopic** or **certified** if it is isotopic and approximates the surface to any given precision. It is known that isotopy is stronger than homeomorphism [2].

Precisely, an **isotopic meshing** for a surface $\mathcal{S} \subset \mathbb{R}^3$ consists of a triangular polyhedron \mathcal{G} and a continuous mapping $\gamma : \mathbb{R}^3 \times [0, 1] \rightarrow \mathbb{R}^3$ which, for any fixed $t \in [0, 1]$, is a homeomorphism $\gamma(\cdot, t)$ from \mathbb{R}^3 to itself, and which continuously deforms \mathcal{G} into \mathcal{S} : $\gamma(\cdot, 0) = id$, $\gamma(\mathcal{G}, 1) = \mathcal{S}$.

An isotopic meshing \mathcal{G} for a surface $\mathcal{S} \subset \mathbb{R}^3$ is called an **ambient isotopic meshing** if, for a given number $\epsilon > 0$, \mathcal{G} gives an ϵ -approximation for \mathcal{S} in the following sense $\|P - \gamma(P, 1)\| \leq \epsilon$ for all $P \in \mathcal{G}$. Such a meshing is also called an **ϵ -meshing**.

We use intervals to isolate real numbers: let $\square\mathbb{Q}$ denote the set of intervals of the form $[a, b]$ where $a < b \in \mathbb{Q}$. The **length** of an interval box $\mathbf{B}_n = [a_1, b_1] \times \cdots \times [a_n, b_n] \in \square\mathbb{Q}^n$ is defined to be $|\mathbf{B}_n| = \max_i(b_i - a_i)$.

Our main result can be summarized as follows.

Theorem 1. *Let $f(x, y, z)$ be a square free polynomial with rational numbers as coefficients. For any algebraic surface $\mathcal{S} : f(x, y, z) = 0$, a given number $\epsilon > 0$, and a bounding box $\mathbf{B} \in \square\mathbb{Q}^3$, we have an algorithm to compute an ϵ -meshing for $\mathcal{S}_{\mathbf{B}} = \mathcal{S} \cap \mathbf{B}$.*

We first give a sketch of our algorithm and will explain some of the key steps later in this extended abstract.

Algorithm 2. $\text{AMeshSur}(f(x, y, z), \mathbf{B}, \epsilon)$. *The input is the same as described in Theorem 7. The output is an ϵ -meshing for $\mathcal{S}_{\mathbf{B}}$.*

- S1** Compute the strong projection curve $\mathcal{C} : g(x, y) = 0$ and an ϵ -meshing for \mathcal{C} .
- S2** Meshing the singular part of \mathcal{S} .
- S3** Meshing the non-singular part of \mathcal{S} .
- S4** Merge the meshes obtained in Steps **S2** and **S3**.

We will explain how to compute the strong projection curve in Section 2. The methods to compute an ϵ -meshing for a plane curve can be found in [4]. We will briefly show how to mesh the singular parts of \mathcal{S} in Section 3. We use modified methods from [3, 11] to mesh the non-singular part of \mathcal{S} . Details of our results can be found in [5, 7, 8].

There exist four main approaches to compute isotopic meshings for surfaces: the marching cube method, the Morse theory method, the Delaunay refinement method, and the CAD (Cylindrical Algebraic Decomposition) based method [2]. Of these methods, only the CAD based methods are capable of treating surfaces with singularities.

We give a method to compute a certified meshing for implicit algebraic surfaces with singularities. The method is a hybrid one based on the CAD approach. We use symbolic computation to guarantee the correctness and use numerical computation whenever possible to enhance the efficiency. To our knowledge, this is the first method to compute an ambient isotopic meshing for surfaces with singularities.

To use a triangular polyhedron to approximate a surface, we usually need polyhedrons with thousands of faces. A strategy to reduce the number of meshes is to use quadratic surfaces to construct certified approximation. This has been done for algebraic curves [9, 10] and is an interest problem for algebraic surfaces.

2 Strong Projection Curve

The basic idea for the CAD based methods to compute the topology of a surface \mathcal{S} is to project \mathcal{S} to the xy -plane, compute the topology of the projection curve, and obtain the topology of \mathcal{S} by lifting the topology of the projection curve to the space. To compute the ambient meshing for a surface, we need a strong projection curve, which will be discussed in this section.

Let $\mathcal{S} : f(x, y, z) = 0$ be an algebraic surface, where $f(x, y, z) \in \mathbb{Q}[x, y, z]$ is square free. A point P_0 is a **z -critical point** of \mathcal{S} if $f(P_0) = f_z(P_0) = 0$.

Let

$$G_1(x, y) = \text{sqrfree}(\text{Res}(f, \frac{\partial f}{\partial z}, z)). \quad (1)$$

The plane curve $G_1(x, y) = 0$ is called the **projection curve** of \mathcal{S} .

A point is called **z -extremal** of a surface or a space curve if the surface or curve achieves a local extremum value at this point in the z -direction. In order to compute the ambient meshing for a surface, we need to project the z -extremal points of \mathcal{S} and the space curve $f(x, y, z) = G_1(x, y) = 0$ to the plane. We have

Lemma 1. *Let $f(x, y, z) = \prod_i f_i(x, y, z)$ be a square free polynomial and f_i irreducible polynomials. A necessary condition for the surface $f(x, y, z) = 0$ to have a z -extremal point is*

$$G_2(x, y) = \prod_i \text{Res}(f_i, \frac{\partial f_i}{\partial x}, z) \prod_i \text{Res}(f_i, \frac{\partial f_i}{\partial y}, z) = 0 \quad (2)$$

where only the nonzero resultants are included.

The following example shows that we need to consider the irreducible factors. Let $f = (z - y)(z - x)(x^2 + y^2 + z^2 - 1)$. Then $\text{Res}(f, f_x, z) = \text{Res}(f, f_y, z) \equiv 0$. But the surface indeed has a z -extremal point at $(0, 0, 1)$.

We also need to consider the z -extremal points of spatial curves defined by $g(x, y) = f(x, y, z) = 0$, where g and f are polynomials. For this purpose, we need to decompose the curve into irreducible ones. The leading coefficient of g (f) as a univariate polynomial in y (z) is called the **initial** of g (f). Any spatial curve $f(x, y, z) = g(x, y) = 0$ can be decomposed into the union of irreducible curves represented by irreducible chains algorithmically [12][5]. The initials of these irreducible chains are univariate polynomial in x . We have:

Lemma 2. *Let $g(x, y), f(x, y, z)$ be an irreducible chain and*

$$\begin{aligned} I(x) &= \text{product of the initials of } f, g. \\ T(x) &= \text{Res}(\text{Res}(h, f, z), g, y) \text{ where } h(x, y, z) = f_x g_y - f_y g_x. \end{aligned} \quad (3)$$

Let E be the set of z -extremal points of the curve $\mathcal{C} : f = g = 0$. Then $\text{Proj}_x(E) \subset V(T(x)) \cup V(I(x))$. Furthermore, if $T(x) \equiv 0$, then the curve defined by f, g is contained in several planes perpendicular to the z -axis.

The following example shows that we need to decompose the curve into irreducible ones. Let $f = z(x^2 + z^2 - 1), g = y$. Then $\text{Res}(f_x g_y - f_y g_x, f, z) \equiv 0$. But the curve indeed has a z -extremal point at $(0, 0, 1)$.

The plane curve

$$g(x, y) = G_1(x, y)G_2(x, y)T(x)I(x) \quad (4)$$

is called the **strong projection curve** of surface \mathcal{S} , where G_1, G_2, T and I are defined in (1), (2), and (3) respectively. In the case of (3), we will include the nonzero projections for all irreducible components of $G_1(x, y)G_2(x, y) = f(x, y, z) = 0$.

The purpose to introduce the concept of strong projection curve is that on a region containing no points of $g(x, y) = 0$, the surface \mathcal{S} and the space curve $g(x, y) = f(x, y, z) = 0$ have no z -extremal points. This property allows us to estimate the z -values of a surface patch or a curve segment over a region R with their values on the boundary of R .

3 Segregating Box for Points and Curve Segments

A key idea to mesh the singular part of $\mathcal{S} : f(x, y, z) = 0$ is to compute the segregating boxes for the critical points and critical curve segments of \mathcal{S} . The idea of segregating boxes for points of plane curves was originally introduced in [1]. Here, we give a new interval based method to compute it and extend the concept to critical curve segments.

The critical parts of \mathcal{S} consists of the **critical curve** defined by $g(x, y) = f(x, y, z) = 0$ and the **critical points** defined by $h(x) = g(x, y) = f(x, y, z) = 0$, where g is defined in (4) and $h(x) = \text{Res}(g, g_y, y)$.

We may compute isolating boxes for the singular points of \mathcal{S} with the method given in [8]. Let $P = (\alpha, \beta, \gamma)$ be a critical point on \mathcal{S} such that $f(\alpha, \beta, z) = 0$ has a finite number of solutions. Then \mathbf{B} is called a **segregating box** of P if \mathcal{S} does not intersect with the top and bottom faces of \mathbf{B} . A segregating box can be computed as follows. We may compute an isolating interval $[u, v]$ of γ as the solution of $f(\alpha, \beta, z) = 0$. Let $\mathbf{B}_2 = [a, b] \times [c, d]$ be a box containing (α, β) . We may compute the inclusion function $\square f(\mathbf{B}_2, u)$, $\square f(\mathbf{B}_2, v)$ and subdivide \mathbf{B}_2 until $0 \notin \square f(\mathbf{B}_2, u)$ and $0 \notin \square f(\mathbf{B}_2, v)$. This process will terminate since $f(\alpha, \beta, u)f(\alpha, \beta, v) \neq 0$.

In a similar way, we may assume that the strong projection curve $\mathcal{C} : g(x, y) = 0$ of \mathcal{S} does not meet the top and bottom of \mathbf{B}_2 . As a consequence, \mathcal{S} intersects \mathbf{B} on the four side faces and the critical curves of \mathcal{S} intersects \mathbf{B} on the left and right faces: $[a, a] \times [c, d] \times [u, v]$ and $[b, b] \times [c, d] \times [u, v]$. With these conditions, it is not difficult to compute the topology and meshing for \mathcal{S} inside \mathbf{B} . We can subdivide the boxes until $|\mathbf{B}| < \epsilon$ and the meshings thus obtained are ϵ -meshings for \mathcal{S} .

Now, we introduce briefly how to mesh \mathcal{S} near a critical curve segment. A box $\mathbf{B} = [a, b] \times [c, d] \times [u, v]$ is called a **segregating box** for a critical curve segment S inside \mathbf{B} if \mathcal{S} is the only smooth curve branch of the critical curve inside \mathbf{B} and S does not intersect with the top and bottom faces of \mathbf{B} .

We may compute a segregating box for a critical curve segment S as follows. Let P_a and P_b be the intersection points of S with planes $x = a$ and $x = b$ respectively. Then we may compute an isolating box $\mathbf{B}_a = [y_{a,1}, y_{a,2}] \times [z_{a,1}, z_{a,2}]$ for P_a by solving the equations $g(a, y) = f(a, y, z) = 0$ with [6]. Compute $\mathbf{B}_b = [y_{b,1}, y_{b,2}] \times [z_{b,1}, z_{b,2}]$ similarly. We may set the segregating box of S to be $\mathbf{B}_P = [a, b] \times [\min\{y_{a,1}, y_{b,1}\}, \max\{y_{a,2}, y_{b,2}\}] \times [\min\{z_{a,1}, z_{b,1}\}, \max\{z_{a,2}, z_{b,2}\}]$. This is

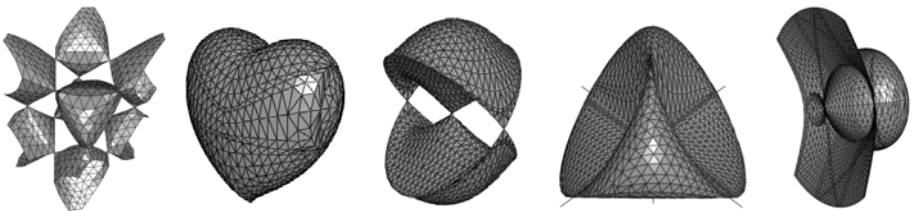


Fig. 1. Meshing for surfaces with singular points and singular curves

true because, due to the conditions in the strong projection curve, S is monotone both in y and z directions. If $|\mathbf{B}_P| > \epsilon$, we can further subdivide the boxes. After the segregating boxes for the critical curve segments are constructed, we can compute a meshing for the surface inside \mathbf{B} .

Now, we have computed the meshing and topological structure for S inside the segregating boxes for the critical point and critical curve of S . Outside these boxes, S has no singular points and can be meshed by modifying the methods in [3,11].

Figure 1 are the meshings for five surfaces computed with the implementation of our algorithm in Maple. Equations defining these surfaces can be found in [5].

References

1. Arnon, D.S., Collins, G., McCallum, S.: Cylindrical Algebraic Decomposition, II: An Adjacency Algorithm for Plane. *SIAM J. on Comput.* 13(4), 878–889 (1984)
2. Boissonnat, J.D., Cohen-Steiner, D., Mourrain, B., Rote, G., Vegter, G.: Meshing of Surfaces. In: *Effective Computational Geometry for Curves and Surfaces*, pp. 181–230. Springer, Berlin (2006)
3. Cheng, S.W., Dey, T.K., Ramos, A., Ray, T.: Sampling and Meshing a Surface with Guaranteed Topology and Geometry. In: *Proc. 20th Symposium on Computational Geometry*, pp. 280–289. ACM Press, New York (2004)
4. Cheng, J.S., Gao, X.S., Li, J.: Topology Determination and Isolation for Implicit Plane Curves. In: *Proc. ACM Symposium on Applied Computing*, pp. 1140–1141 (2009)
5. Cheng, J.S., Gao, X.S., Li, J.: Ambient Isotopic Meshing of Implicit Algebraic Surface with Singularities. *MM-Preprints* 27, 150–183 (2008) Arxiv preprint arXiv:0903.3524 (2009)
6. Cheng, J.S., Gao, X.S., Li, J.: Root Isolation for Bivariate Polynomial Systems with Local Generic Position Method (2009) (accepted ACM ISSAC)
7. Cheng, J.S., Gao, X.S., Li, M.: Determine the Topology of Real Algebraic Surfaces. In: Martin, R., Bez, H.E., Sabin, M.A. (eds.) *IMA 2005. LNCS*, vol. 3604, pp. 121–146. Springer, Heidelberg (2005)
8. Cheng, J.S., Gao, X.S., Yap, C.K.: Complete Numerical Isolation of Real Roots in Zero-dimensional Triangular Systems. *Journal of Symbolic Computation* 44(7), 768–785 (2009)
9. Gao, X.S., Li, M.: Rational Quadratic Approximation to Real Algebraic Curves. *Computer Aided Geometric Design* 21, 805–828 (2004)
10. Li, M., Gao, X.S., Chou, S.C.: Quadratic Approximation to Plane Parametric Curves and Applications in Approximate Implicitization. *Visual Computers* 22, 906–917 (2006)
11. Plantinga, S., Vegter, G.: Isotopic Meshing of Implicit Surfaces. *Visual Computer* 23, 45–58 (2007)
12. Wu, W.T.: *Mathematics Machenization*. Science Press/Kluwer, Beijing (2001)

Involution and Difference Schemes for the Navier–Stokes Equations

Vladimir P. Gerdt¹ and Yuri A. Blinkov²

¹ Laboratory of Information Technologies, Joint Institute for Nuclear Research,
141980 Dubna, Russia

gerdt@jinr.ru

² Department of Mathematics and Mechanics, Saratov State University
410012 Saratov, Russia

BlinkovUA@info.sgu.ru

Abstract. In the present paper we consider the Navier–Stokes equations for the two-dimensional viscous incompressible fluid flows and apply to these equations our earlier designed general algorithmic approach to generation of finite-difference schemes. In doing so, we complete first the Navier–Stokes equations to involution by computing their Janet basis and discretize this basis by its conversion into the integral conservation law form. Then we again complete the obtained difference system to involution with eliminating the partial derivatives and extracting the minimal Gröbner basis from the Janet basis. The elements in the obtained difference Gröbner basis that do not contain partial derivatives of the dependent variables compose a conservative difference scheme. By exploiting arbitrariness in the numerical integration approximation we derive two finite-difference schemes that are similar to the classical scheme by Harlow and Welch. Each of the two schemes is characterized by a 5×5 stencil on an orthogonal and uniform grid. We also demonstrate how an inconsistent difference scheme with a 3×3 stencil is generated by an inappropriate numerical approximation of the underlying integrals.

1 Introduction

In this paper we consider the Navier–Stokes equations for the Newtonian incompressible fluids with constant viscosity. We restrict ourselves to two-dimensional flows though all the below results admit a straightforward extension to the three-dimensional case. The incompressibility and constancy of viscosity are assumed in most mathematical treatments of the two- and three-dimensional Navier–Stokes equations (see, for example, [1]), since it is a common belief that such simplification of the equations still preserves their applicability to description of the main features of laminar and turbulent flows. Though in this paper we do not impose any restriction on the Reynolds number and treat this number as a parameter, the difference schemes we obtain are not expected to be appropriate to turbulent flows. Thus, it is implicitly assumed that the Reynolds number is small enough.

To generate a finite-difference scheme for the Navier–Stokes equations we apply our general algorithmic approach [2] based on the integral conservation law form of the initial system extended with some relevant integral relations between the dependent variables and their partial derivatives. Then discretization of the system is done by choosing an integration contour (or surface in the three-dimensional case) on an appropriate grid, and a finite-difference scheme is obtained by the (difference) elimination of the partial derivatives.

In paper [2] we considered differential systems of the Cauchy–Kovalevskaya type that is trivially involutive. Generally, before discretizing, a system of differential equations has to be completed to involution. More precisely, the system has to be completed to its formally integrable form, that is, such that all the integrability conditions are incorporated into the system. The integrability conditions are differential consequences of the system of the order not higher than the order of the system and such that they cannot be obtained from the system by means of pure algebraic transformations, i.e. without differentiation.

A constructive way to compute the integrability conditions and to incorporate them into the system is to complete the system to involution by doing prolongation of equations in the system - taking their derivatives with respect to the independent variables - and elimination of the highest order derivatives from the prolonged equations [3] (see also the introductory paper [4]). Without completion of a differential system to involution some hidden integrability conditions may not be satisfied in the discrete version within a reasonable accuracy. If so a numerical solution does not preserve internal algebraic and differential properties of the continuous solution.

Furthermore, an inappropriate discretization may lead to a difference scheme which is inconsistent with the initial differential system. The inconsistency means the existence of a difference consequence of the discrete system that in the continuous limit becomes a differential equation which does not follow from the initial differential system.

The structure of the paper is as follows. In Section 2 we consider the Navier–Stokes equations in the Cartesian coordinates and complete them to involution using the Janet monomial division [5,6,7]. In doing so we detect just one integrability condition, namely, the pressure Poisson equation. In Section 3 we discretize the involutive Navier–Stokes system by applying the approach of paper [2] based on the integral form of the system. The structure of the discrete system depends on numerical approximation methods for the integrals. Here we consider two simple approximation methods that lead to two distinct discrete systems. Then in Section 4 we construct for the both discrete systems the minimal Gröbner bases by extracting them from the corresponding Janet bases for the ranking that eliminates the partial derivatives and, thus, can serve to generate finite-difference schemes for the Navier–Stokes equations. The schemes generated in this way are qualitatively similar to the classical scheme derived by Harlow and Welch in [8]. In Section 5 we discuss consistency of the discrete version of the Navier–Stokes equations with their differential form. By explicit computation we demonstrate that an inappropriate numerical integration may lead to a difference

scheme whose algebraically rigorous difference consequence yields in the continuous limit a differential equation which does not follow from the Navier–Stokes equations. Thereby such a scheme is inconsistent. We conclude in Section 6.

2 Involution Form of the Navier–Stokes Equations

We consider unsteady two-dimensional motion of incompressible viscous liquid of constant viscosity that is governed by the following system of equations which we refer to as the Navier–Stokes system:

$$\begin{cases} f^1 := u_x + v_y = 0, \\ f^2 := u_t + uu_x + vv_y = -p_x + \frac{1}{\text{Re}}\Delta u, \\ f^3 := v_t + uv_x + vv_y = -p_y + \frac{1}{\text{Re}}\Delta v. \end{cases} \quad (1)$$

Here f^1 is the continuity equation, and f^1 and f^2 are the proper Navier–Stokes equations [1]. Equations in (1) are written in the dimensionless form, where (u, v) is the velocity field, and p is the pressure. The density is included in the Reynolds number denoted by Re . To determine a nonstationary flow one has to specify the initial data $u = u_0(x, y)$ and $v = v_0(x, y)$ satisfying equation f^1 . The boundary conditions on a solid surface lead to vanishing relative speed of the liquid. For the pressure there is no need to specify the boundary conditions on the surface. Throughout this paper we use notions and definitions in (1) such that $x \succ y \succ t$ and $u \succ v \succ p$, then completion of the system (1) to involution based on the Janet division reveals the only integrability condition

$$f^4 := u_x^2 + 2v_xu_y + v_y^2 = -\Delta p$$

which is the differential consequence of the equations in (1)

$$f_x^2 + f_y^3 - f_t^1 - uf_x^1 - vf_y^1 + \frac{1}{\text{Re}}\Delta f^1 = f^4. \quad (2)$$

Therefore, the involutive Janet form of the Navier–Stokes system is given by

$$\begin{cases} f^1 : u_x + v_y = 0, \\ f^2 : u_t + uu_x + vv_y = -p_x + \frac{1}{\text{Re}}\Delta u, \\ f^3 : v_t + uv_x + vv_y = -p_y + \frac{1}{\text{Re}}\Delta v, \\ f^4 : u_x^2 + 2v_xu_y + v_y^2 = -\Delta p. \end{cases} \quad (3)$$

Equation (2) which plays an important role in the computational fluid dynamics is usually called the pressure Poisson equation [9]. By completion of the Navier–Stokes system to involution this equation was obtained first by applying Cartan’s algorithm [10]. Apparently, the pressure Poisson equation is to be detected by any completion procedure. For instance, in [4] it was obtained by the geometric Cartan–Kuranishi completion.

To see that the system (3) is in involution, and hence there are no other integrability conditions, we give in the below table the following data for the equations in the system: the first column enumerates the equations; the second column shows the leader, i.e. the highest ranking derivative in the equation; the third and the fourth columns show, respectively, Janet multiplicative and nonmultiplicative independent variables for the equations.

equation	leader	multiplicative	nonmultiplicative
f^1	u_x	$\{x, y, t\}$	\emptyset
f^2	u_{yy}	$\{y, t\}$	$\{x\}$
f^3	v_{xx}	$\{x, y, t\}$	\emptyset
f^4	p_{xx}	$\{x, y, t\}$	\emptyset

Thus, only f^2 has a nonmultiplicative prolongation, namely, the prolongation with respect to x . Equality (2) explicitly demonstrates that this prolongation is reduced to zero by the multiplicative prolongations of the other equations in (3). This is just the condition of involutivity [5,7] for the system.

Apparently, the system of equations (3) is a formally integrable extension of (1) of the minimal cardinality.

3 Discretization

The pressure Poisson equation f^4 in (3) can be written as

$$f^4 := \frac{\partial^2}{\partial x^2} (u^2 + p) + \frac{\partial^2}{\partial x \partial y} (2uv) + \frac{\partial^2}{\partial y^2} (v^2 + p) = 0,$$

and, thus, the involutive differential system (3) admits the conservation law form:

$$\begin{cases} f^1 : \frac{\partial}{\partial x} u + \frac{\partial}{\partial y} v = 0, \\ f^2 : \frac{\partial}{\partial t} u + \frac{\partial}{\partial x} (u^2 + p - \frac{1}{\text{Re}} u_x) + \frac{\partial}{\partial y} (vu - \frac{1}{\text{Re}} u_y) = 0, \\ f^3 : \frac{\partial}{\partial t} v + \frac{\partial}{\partial x} (uv - \frac{1}{\text{Re}} v_x) + \frac{\partial}{\partial y} (v^2 + p - \frac{1}{\text{Re}} v_y) = 0, \\ f^4 : \frac{\partial}{\partial x} (u u_x + v u_y + p_x) + \frac{\partial}{\partial y} (v v_y + u v_x + p_y) = 0. \end{cases} \quad (4)$$

Consider now the square integration contour Γ in the plane (x, y) as shown in Fig. 1 and convert system (4) into the equivalent integral system:

$$\begin{cases} \oint_{\Gamma} -v dx + u dy = 0, \\ \int_{x_j}^{x_{j+2}} \int_{y_k}^{y_{k+2}} u dx dy \Big|_{t_n}^{t_{n+1}} - \int_{t_n}^{t_{n+1}} \left(\oint_{\Gamma} (vu - \frac{1}{\text{Re}} u_y) dx - (u^2 + p - \frac{1}{\text{Re}} u_x) dy \right) dt = 0, \\ \int_{x_j}^{x_{j+2}} \int_{y_k}^{y_{k+2}} v dx dy \Big|_{t_n}^{t_{n+1}} - \int_{t_n}^{t_{n+1}} \left(\oint_{\Gamma} (v^2 + p - \frac{1}{\text{Re}} v_y) dx - (uv - \frac{1}{\text{Re}} v_x) dy \right) dt = 0, \\ \oint_{\Gamma} -((v^2)_y + (uv)_x + p_y) dx + ((u^2)_x + (vu)_y + p_x) dy = 0. \end{cases} \quad (5)$$

To pass to the discrete form of (5) we follow our approach in [2] and consider the orthogonal and uniform grids with the mesh steps h in x and y and τ in t , i.e.

$$x_{j+1} - x_j = y_{k+1} - y_k = h, \quad t_{m+1} - t_m = \tau.$$

Now we add to system (5) the integral relations between dependent variables and all their partial derivatives entering into the system:

$$\left\{ \begin{array}{l} \int_{x_j}^{x_{j+1}} (u^2)_x dx = u(x_{j+1}, y)^2 - u(x_j, y)^2, \\ \int_{y_k}^{y_{k+1}} (v^2)_y dy = v(x, y_{k+1})^2 - v(x, y_k)^2, \\ \int_{x_j}^{x_{j+1}} (uv)_x dx = u(x_{j+1}, y)v(x_{j+1}, y) - u(x_j, y)v(x_j, y), \\ \int_{y_k}^{y_{k+1}} (uv)_y dy = u(x, y_{k+1})v(x, y_{k+1}) - u(x, y_k)v(x, y_k), \\ \int_{x_j}^{x_{j+1}} u_x dx = u(x_{j+1}, y) - u(x_j, y), \quad \int_{y_k}^{y_{k+1}} u_y dy = u(x, y_{k+1}) - u(x, y_k), \\ \int_{x_j}^{x_{j+1}} v_x dx = v(x_{j+1}, y) - v(x_j, y), \quad \int_{y_k}^{y_{k+1}} v_y dy = v(x, y_{k+1}) - v(x, y_k), \\ \int_{x_j}^{x_{j+1}} p_x dx = p(x_{j+1}, y) - p(x_j, y), \quad \int_{y_k}^{y_{k+1}} p_y dy = p(x, y_{k+1}) - p(x, y_k). \end{array} \right. \quad (6)$$

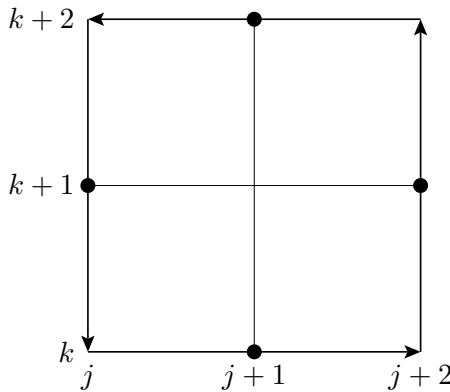


Fig. 1. Integration contour

By applying the midpoint rule to approximate integration over x and y and the rectangle rule for integration over t in the system (5.6), we obtain the difference equations on the grid:

$$\left\{ \begin{array}{l}
 (u^2)_{x_{j+1k}}^n \cdot 2h = u_{j+2k}^n{}^2 - u_{jk}^n{}^2, \\
 (v^2)_{y_{jk+1}}^n \cdot 2h = v_{j+2k}^n{}^2 - v_{jk}^n{}^2, \\
 (uv)_{x_{j+1k}}^n \cdot 2h = u_{j+2k}^n v_{j+2k}^n - u_{jk}^n v_{jk}^n, \\
 (uv)_{y_{jk+1}}^n \cdot 2h = u_{j+2k}^n v_{j+2k}^n - u_{jk}^n v_{jk}^n, \\
 u_{x_{j+1k}}^n \cdot 2h = u_{j+2k}^n - u_{jk}^n, \\
 u_{y_{jk+1}}^n \cdot 2h = u_{j+2k}^n - u_{jk}^n, \\
 v_{x_{j+1k}}^n \cdot 2h = v_{j+2k}^n - v_{jk}^n, \\
 v_{y_{jk+1}}^n \cdot 2h = v_{j+2k}^n - v_{jk}^n, \\
 p_{x_{j+1k}}^n \cdot 2h = p_{j+2k}^n - p_{jk}^n, \\
 p_{y_{jk+1}}^n \cdot 2h = p_{j+2k}^n - p_{jk}^n, \\
 -(v_{j+1k}^n - v_{j+1k+2}^n) \cdot 2h + (u_{j+2k+1}^n - u_{j+1k}^n) \cdot 2h = 0, \\
 (u_{j+1k+1}^{n+1} - u_{j+1k+1}^n) \cdot 4h^2 - 2h\tau \left(v_{j+1k}^n u_{j+1k}^n + \frac{1}{\text{Re}} u_{y_{j+1k}}^n + \right. \\
 \quad \left. + v_{j+1k+2}^n u_{j+1k+2}^n - \frac{1}{\text{Re}} u_{y_{j+1k+2}}^n - (u^2)_{j+2k+1}^n - p_{j+2k+1}^n + \right. \\
 \quad \left. + \frac{1}{\text{Re}} u_{x_{j+2k+1}}^n + (u^2)_{j+1k+1}^n + p_{j+1k+1}^n - \frac{1}{\text{Re}} u_{x_{j+1k+1}}^n \right) = 0, \\
 (v_{j+1k+1}^{n+1} - v_{j+1k+1}^n) \cdot 4h^2 - 2h\tau \left((v^2)_{j+1k}^n - p_{j+1k}^n \frac{1}{\text{Re}} v_{y_{j+1k}}^n + \right. \\
 \quad \left. + (v^2)_{j+1k+2}^n + p_{j+1k+2}^n - \frac{1}{\text{Re}} v_{y_{j+1k+2}}^n - \right. \\
 \quad \left. - u_{j+2k+1}^n v_{j+2k+1}^n + \frac{1}{\text{Re}} v_{x_{j+2k+1}}^n + u_{j+1k+1}^n v_{j+1k+1}^n + \frac{1}{\text{Re}} v_{x_{j+1k+1}}^n \right) = 0, \\
 (v^2)_{y_{j+1k+2}}^n + (uv)_{x_{j+1k+2}}^n + p_{y_{j+1k+2}}^n - (v^2)_{y_{j+1k}}^n - \\
 \quad - (uv)_{x_{j+1k+2}}^n - p_{y_{j+1k+2}}^n + (u^2)_{x_{j+2k+1}}^n + \\
 \quad + (uv)_{y_{j+2k+1}}^n - p_{x_{j+2k+1}}^n - (u^2)_{x_{j+1k+1}}^n - (uv)_{y_{j+1k+1}}^n - p_{x_{j+1k+1}}^n = 0.
 \end{array} \right. \quad (7)$$

Hereafter we use the operator notations to represent the difference equations in a more compact form. Denote by θ_α ($\alpha \in \{x, y, t\}$) the right-shift operator in the variable α , by $I_{r\alpha}$ the integration operator over the variable α and by D_1 and D_2 , respectively, the difference and two-step difference operators. Then the following operator relations hold

$$\begin{aligned}
 I_{rx} &= 2h\theta_x, & I_{ry} &= 2h\theta_y, & I_{rxy} &= 4h^2\theta_x\theta_y, \\
 I_{rt} &= \tau, & D_{1\alpha} &= \theta_\alpha - 1, & D_{2\alpha} &= \theta_\alpha^2 - 1,
 \end{aligned} \quad (8)$$

and the difference system (7) reads

$$\left\{ \begin{array}{l} I_{rx} \circ (u^2)_x = D_{2x} \circ u^2, \quad I_{rx} \circ (vu)_x = D_{2x} \circ vu, \\ I_{ry} \circ (uv)_y = D_{2y} \circ uv, \quad I_{ry} \circ (v^2)_y = D_{2y} \circ v^2, \\ I_{rx} \circ u_x = D_{2x} \circ u, \quad I_{ry} \circ u_y = D_{2y} \circ u, \\ I_{rx} \circ v_x = D_{2x} \circ v, \quad I_{ry} \circ v_y = D_{2y} \circ v, \\ I_{rx} \circ p_x = D_{2x} \circ p, \quad I_{ry} \circ p_y = D_{2y} \circ p, \\ I_{ry} D_{2x} \circ u + I_{rx} D_{2y} \circ v = 0, \\ I_{rxy} D_{1t} \circ u + I_{rt} (I_{rx} D_{2y} \circ (vu - \frac{1}{\text{Re}} u_y) + I_{ry} D_{2x} \circ (u^2 + p - \frac{1}{\text{Re}} u_x)) = 0, \\ I_{rxy} D_{1t} \circ v + I_{rt} (I_{rx} D_{2y} \circ (v^2 + p - \frac{1}{\text{Re}} v_y) + I_{ry} D_{2x} \circ (uv - \frac{1}{\text{Re}} v_x)) = 0, \\ I_{rx} D_{2y} \circ ((v^2)_y + (uv)_x + p_y) + I_{ry} D_{2x} \circ ((u^2)_x + (vu)_y + p_x) = 0. \end{array} \right. \quad (9)$$

As an example of another possible discretization we consider application of the trapezoidal rule to the integral relations for u_x, u_y, v_x, v_y in (6). For this case we shall use the corresponding integration operators

$$I_{tx} = \frac{h}{2}(\theta_x + 1), \quad I_{ty} = \frac{h}{2}(\theta_y + 1) \quad (10)$$

what leads to the following modification of (9)

$$\left\{ \begin{array}{l} I_{rx} \circ (u^2)_x = D_{2x} \circ u^2, \quad I_{rx} \circ (vu)_x = D_{2x} \circ vu, \\ I_{ry} \circ (uv)_y = D_{2y} \circ uv, \quad I_{ry} \circ (v^2)_y = D_{2y} \circ v^2, \\ I_{tx} \circ u_x = D_{1x} \circ u, \quad I_{ty} \circ u_y = D_{1y} \circ u, \\ I_{tx} \circ v_x = D_{1x} \circ v, \quad I_{ty} \circ v_y = D_{1y} \circ v, \\ I_{rx} \circ p_x = D_{2x} \circ p, \quad I_{ry} \circ p_y = D_{2y} \circ p, \\ I_{ry} D_{2x} \circ u + I_{rx} D_{2y} \circ v = 0, \\ I_{rxy} D_{1t} \circ u + I_{rt} (I_{rx} D_{2y} \circ (vu - \frac{1}{\text{Re}} u_y) + I_{ry} D_{2x} \circ (u^2 + p - \frac{1}{\text{Re}} u_x)) = 0, \\ I_{rxy} D_{1t} \circ v + I_{rt} (I_{rx} D_{2y} \circ (v^2 + p - \frac{1}{\text{Re}} v_y) + I_{ry} D_{2x} \circ (uv - \frac{1}{\text{Re}} v_x)) = 0, \\ I_{rx} D_{2y} \circ ((v^2)_y + (uv)_x + p_y) + I_{ry} D_{2x} \circ ((u^2)_x + (vu)_y + p_x) = 0. \end{array} \right. \quad (11)$$

4 Difference Elimination of Partial Derivatives

In order to eliminate partial derivatives from (9) and (11) and hence to obtain difference schemes we construct difference Gröbner bases [2] (for a more rigorous treatment of Gröbner bases in the difference polynomial rings, cf. [11]) for these systems under the elimination ranking

$$u_x \succ u_y \succ v_x \succ v_y \succ u \succ v \succ p, \quad \theta_t \succ \theta_x \succ \theta_y. \quad (12)$$

Since there is no software available for computing Gröbner bases for nonlinear difference polynomials, we performed the computation “by hand”. In doing so, we used the involutive completion procedure based on the Janet division [2]. The obtained Janet basis as a redundant Gröbner basis is rather cumbersome. For this reason we give below only its subset which is a minimal Gröbner basis. The basis is not reduced.¹ We use this redundant form because it is much more

¹ The last term in the equation e_2 in (13) is redundant modulo e_1 .

compact than the reduced Gröbner basis and explicitly shows its connection to the differential (3) and difference (11) systems.

$$\left\{ \begin{array}{l}
 I_{rx} \circ (u^2)_x = D_{2x} \circ u^2, \quad I_{rx} \circ (vu)_x = D_{2x} \circ vu, \\
 I_{ry} \circ (uv)_y = D_{2y} \circ uv, \quad I_{ry} \circ (v^2)_y = D_{2y} \circ v^2, \\
 I_{rx} \circ u_x = D_{2x} \circ u, \quad I_{ry} \circ u_y = D_{2y} \circ u, \\
 I_{rx} \circ v_x = D_{2x} \circ v, \quad I_{ry} \circ v_y = D_{2y} \circ v, \\
 I_{rx} \circ p_x = D_{2x} \circ p, \quad I_{ry} \circ p_y = D_{2y} \circ p, \\
 e_1 := I_{ry} D_{2x} \circ u + I_{rx} D_{2y} \circ v = 0, \\
 e_2 := I_{rx} I_{ry} I_{rxy} D_{1t} \circ u + I_{rt} (I_{rx} I_{ry}^2 D_{2x} \circ (u^2 + p) + \\
 \quad + I_{rx}^2 I_{ry} D_{2y} \circ (vu) - \frac{1}{\text{Re}} (I_{ry}^2 D_{2x}^2 + I_{rx}^2 D_{2y}^2) \circ u) = 0, \\
 I_{rxy} D_{1t} \circ u + I_{rt} (I_{rx} D_{2y} \circ (vu - \frac{1}{\text{Re}} u_y) + I_{ry} D_{2x} \circ (u^2 + p - \frac{1}{\text{Re}} u_x)) = 0, \\
 e_3 := I_{rx} I_{ry} I_{rxy} D_{1t} \circ v + I_{rt} (I_{rx} I_{ry}^2 D_{2x} \circ (uv) + \\
 \quad + I_{rx}^2 I_{ry} D_{2y} \circ (v^2 + p) - \frac{1}{\text{Re}} (I_{ry}^2 D_{2x}^2 + I_{rx}^2 D_{2y}^2) \circ v) = 0, \\
 I_{rxy} D_{1t} \circ v + I_{rt} (I_{rx} D_{2y} \circ (v^2 + p - \frac{1}{\text{Re}} v_y) + I_{ry} D_{2x} \circ (uv - \frac{1}{\text{Re}} v_x)) = 0, \\
 e_4 := I_{rx}^2 D_{2y}^2 \circ (v^2) + 2I_{rx} I_{ry} D_{2x} D_{2y} \circ (uv) + I_{ry}^2 D_{2x}^2 \circ (u^2) + \\
 \quad + (I_{ry}^2 D_{2x}^2 + I_{rx}^2 D_{2y}^2) \circ p = 0, \\
 I_{rx} D_{2y} \circ ((v^2)_y + (uv)_x + p_y) + I_{ry} D_{2x} \circ ((u^2)_x + (vu)_y + p_x) = 0.
 \end{array} \right. \quad (13)$$

To see that (13) is a Gröbner basis, we note that there is the only critical pair $\{e_1, e_2\}$ to be verified. The corresponding difference S -polynomial $S(e_1, e_2) := I_{rx} I_{ry} I_{rxy} D_{1t} \circ e_1 - I_{ry} D_{2x} \circ e_2$ is reduced to zero modulo (13) in accordance with the equality

$$S(e_1, e_2) = \frac{1}{\text{Re}} I_{rt} (I_{ry}^2 D_{2x}^2 + I_{rx}^2 D_{2y}^2) \circ e_1 + I_{rx} D_{2y} \circ e_3 - I_{rt} \circ e_4. \quad (14)$$

Similarly, a Gröbner basis for the second discrete system (11) is given by

$$\left\{ \begin{array}{l}
 I_{rx} \circ (u^2)_x = D_{2x} \circ u^2, \quad I_{rx} \circ (vu)_x = D_{2x} \circ vu, \\
 I_{ry} \circ (uv)_y = D_{2y} \circ uv, \quad I_{ry} \circ (v^2)_y = D_{2y} \circ v^2, \\
 I_{tx} \circ u_x = D_{1x} \circ u, \quad I_{ty} \circ u_y = D_{1y} \circ u, \\
 I_{tx} \circ v_x = D_{1x} \circ v, \quad I_{ty} \circ v_y = D_{1y} \circ v, \\
 I_{rx} \circ p_x = D_{2x} \circ p, \quad I_{ry} \circ p_y = D_{2y} \circ p, \\
 e_1 := I_{ry} D_{2x} \circ u + I_{rx} D_{2y} \circ v = 0, \\
 e_2 := I_{rxy} D_{1t} \circ u + I_{rt} (I_{rx} D_{2y} \circ (vu) + \\
 \quad + I_{ry} D_{2x} \circ (u^2 + p) - \frac{2}{h} \frac{1}{\text{Re}} (I_{ry} D_{1x}^2 + I_{rx} D_{1y}^2) \circ u) = 0, \\
 I_{rxy} D_{1t} \circ u + I_{rt} (I_{rx} D_{2y} \circ (vu - \frac{1}{\text{Re}} u_y) + I_{ry} D_{2x} \circ (u^2 + p - \frac{1}{\text{Re}} u_x)) = 0, \\
 e_3 := I_{rxy} D_{1t} \circ v + I_{rt} (I_{rx} D_{2y} \circ (uv) + \\
 \quad + I_{ry} D_{2x} \circ (v^2 + p) - \frac{2}{h} \frac{1}{\text{Re}} (I_{ry} D_{1x}^2 + I_{rx} D_{1y}^2) \circ v) = 0, \\
 I_{rxy} D_{1t} \circ v + I_{rt} (I_{rx} D_{2y} \circ (v^2 + p - \frac{1}{\text{Re}} v_y) + I_{ry} D_{2x} \circ (uv - \frac{1}{\text{Re}} v_x)) = 0, \\
 e_4 := I_{rx}^2 D_{2y}^2 \circ (v^2) + 2I_{rx} I_{ry} D_{2x} D_{2y} \circ (uv) + I_{ry}^2 D_{2x}^2 \circ (u^2) + \\
 \quad + (I_{ry}^2 D_{2x}^2 + I_{rx}^2 D_{2y}^2) \circ p = 0, \\
 I_{rx} D_{2y} \circ ((v^2)_y + (uv)_x + p_y) + I_{ry} D_{2x} \circ ((u^2)_x + (vu)_y + p_x) = 0.
 \end{array} \right. \quad (15)$$

Again, the equality

$$\begin{aligned}
 I_{r_x} I_{r_y} I_{r_{xy}} D_t \circ e_1 - I_{r_y} D_{2x} \circ e_2 &= \\
 &= \frac{2}{h} \frac{1}{\text{Re}} I_{r_t} (I_{r_y} D_{1x}^2 + I_{r_x} D_{1y}^2) \circ e_1 + I_{r_x} D_{2y} \circ s_3 - I_{r_t} \circ e_4. \quad (16)
 \end{aligned}$$

implies that the left-hand side of (16) - the only difference S -polynomial to be verified - is reduced to zero.

Both systems (13) and (15) include the difference equations e_1, e_2, e_3, e_4 which do not contain partial derivatives of the dependent variables. These equations are just the finite-difference schemes for the involutive Navier–Stokes differential system (3). One can see that the schemes in (13) and (15) have identical equations e_1 and e_4 which are discrete versions of the continuity equation and the pressure Poisson equation, respectively, and the distinct equations e_2 and e_3 discretizing the Navier–Stokes equations.

It is important to emphasize that e_2 in (13) was obtained from the prolongation of the equation that follows e_2 in (13) by action of the operator $I_{r_x} I_{r_y}$ at this equation. The equation e_2 is resulted by reduction of the prolonged equations modulo other equations in the system. The other elements (e_3, e_4) in the difference scheme are also obtained by the prolongation and reduction of the equations that follow e_3 and e_4 in (13). System (15) is derived in much the same way.

The derived difference schemes are qualitatively similar to the classical scheme by Harlow and Welch [8]. All three schemes have the following common features:

- The initial conditions must satisfy the continuity equation f^1 in (1) and the pressure Poisson equation f^4 in (3).
- Transition to the next temporal layer is done by means of the equations e_2 and e_3 that are difference analogues of the Navier–Stokes equations f^2 and f^3 in (1).
- The continuity equation on the next temporal layer is automatically satisfied when the pressure satisfies the discrete version of equation f^4 .

However, as opposed to the approach in [8], we perform first the discrete approximation of system (3) and then verify its consistency (see the next section for more detailed discussion of the consistency check) by means of the equalities (14) and (16). In formula (7) of the paper [8], a discrete version of f^4 was obtained from the consistency condition for the difference scheme. In the last case the derived discrete version of f^4 includes (discrete) partial derivatives of the velocity field (u, v) with respect to time and by this reason looks like much more cumbersome than our equation e_4 .

It should be noted that, by the construction, the schemes in (13) and (15) are conservative since they approximate the integral form (5) of the involutive Navier–Stokes system (3).

5 Consistency Issues

The obtained schemes in (13) and (15) have a 5×5 stencil what is clear from the equations e_2, e_3, e_4 and e_4 , respectively. This involves some difficulties in using those schemes at approaching to boundaries and makes the related numerical computation rather cumbersome. It is tempting to derive a difference scheme with a more compact stencil. Consider the following discretization of (3) which differs from (15) by applying the trapezoidal rule rather than of the midpoint rule for the pressure integral relations:

$$\left\{ \begin{array}{l} I_{tx} \circ (u^2)_x = D_{1x} \circ u^2, \quad I_{rx} \circ (vu)_x = D_{2x} \circ vu, \\ I_{ry} \circ (uv)_y = D_{2y} \circ uv, \quad I_{ty} \circ (v^2)_y = D_{1y} \circ v^2, \\ I_{tx} \circ u_x = D_{1x} \circ u, \quad I_{ty} \circ u_y = D_{1y} \circ u, \\ I_{tx} \circ v_x = D_{1x} \circ v, \quad I_{ty} \circ v_y = D_{1y} \circ v, \\ I_{tx} \circ p_x = D_{1x} \circ p, \quad I_{ty} \circ p_y = D_{1y} \circ p, \\ I_{ry} D_{2x} \circ u + I_{rx} D_{2y} \circ v = 0, \\ I_{rxy} D_{1t} \circ u + I_{rt} (I_{rx} D_{2y} \circ (vu - \frac{1}{\text{Re}} u_y) + I_{ry} D_{2x} \circ (u^2 + p - \frac{1}{\text{Re}} u_x)) = 0, \\ I_{rxy} D_{1t} \circ v + I_{rt} (I_{rx} D_{2y} \circ (v^2 + p - \frac{1}{\text{Re}} v_y) + I_{ry} D_{2x} \circ (uv - \frac{1}{\text{Re}} v_x)) = 0, \\ I_{rx} D_{2y} \circ ((v^2)_y + (uv)_x + p_y) + I_{ry} D_{2x} \circ ((u^2)_x + (vu)_y + p_x) = 0. \end{array} \right. \quad (17)$$

To construct a Gröbner basis for (17) under the elimination ranking (12) we perform, as done in Section 4, suitable prolongations of the discrete versions of the Navier–Stokes equations and the pressure Poisson equation to eliminate the partial derivatives in the independent variables. Then we obtain again the difference scheme consisting of the four difference equations denoted as above by e_1, e_2, e_3, e_4 and entering in the following difference set

$$\left\{ \begin{array}{l} I_{tx} \circ (u^2)_x = D_{1x} \circ u^2, \quad I_{rx} \circ (vu)_x = D_{2x} \circ vu, \\ I_{ry} \circ (uv)_y = D_{2y} \circ uv, \quad I_{ty} \circ (v^2)_y = D_{1x} \circ v^2, \\ I_{tx} \circ u_x = D_{1x} \circ u, \quad I_{ty} \circ u_y = D_{1y} \circ u, \\ I_{tx} \circ v_x = D_{1x} \circ v, \quad I_{ty} \circ v_y = D_{1y} \circ v, \\ I_{tx} \circ p_x = D_{1x} \circ p, \quad I_{ty} \circ p_y = D_{1y} \circ p, \\ e_1 := I_{ry} D_{2x} \circ u + I_{rx} D_{2y} \circ v = 0, \\ e_2 := I_{rxy} D_{1t} \circ u + I_{rt} (I_{rx} D_{2y} \circ (vu) + \\ \quad + I_{ry} D_{2x} \circ (u^2 + p) - \frac{2}{h} \frac{1}{\text{Re}} (I_{ry} D_{1x}^2 + I_{rx} D_{1y}^2) \circ u) = 0, \\ I_{rxy} D_{1t} \circ u + I_{rt} (I_{rx} D_{2y} \circ (vu - \frac{1}{\text{Re}} u_y) + I_{ry} D_{2x} \circ (u^2 + p - \frac{1}{\text{Re}} u_x)) = 0, \\ e_3 := I_{rxy} D_{1t} \circ v + I_{rt} (I_{rx} D_{2y} \circ (uv) + \\ \quad + I_{ry} D_{2x} \circ (v^2 + p) - \frac{2}{h} \frac{1}{\text{Re}} (I_{ry} D_{1x}^2 + I_{rx} D_{1y}^2) \circ v) = 0, \\ I_{rxy} D_{1t} \circ v + I_{rt} (I_{rx} D_{2y} \circ (v^2 + p - \frac{1}{\text{Re}} v_y) + I_{ry} D_{2x} \circ (uv - \frac{1}{\text{Re}} v_x)) = 0, \\ e_4 := \frac{2}{h} I_{rx} D_{1y}^2 \circ (v^2) + 2D_{2x} D_{2y} \circ (uv) + \frac{2}{h} I_{ry} D_{1x}^2 \circ (u^2) + \\ \quad + \frac{2}{h} (I_{ry} D_{1x}^2 + I_{rx} D_{1y}^2) \circ p = 0, \\ I_{rx} D_{2y} \circ ((v^2)_y + (uv)_x + p_y) + I_{ry} D_{2x} \circ ((u^2)_x + (vu)_y + p_x) = 0. \end{array} \right. \quad (18)$$

It is readily seen that the difference scheme e_1, e_2, e_3, e_4 in (18) has a 3×3 stencil. However, the equation set (18), unlike those in (13) and (15), is not a Gröbner basis. To show this we construct the difference S -polynomial $S(e_1, e_2) := I_{r_{xy}} D_t \circ e_1 - I_{r_y} D_{2_x} \circ e_2$. Its reduction modulo e_1, e_2, e_3, e_4 reads

$$S(e_1, e_2) = \frac{2}{h} \frac{1}{\text{Re}} I_{rt} I_{rx} I_{ry} (I_{ry} D_{1_x}^2 + I_{rx} D_{1_y}^2) \circ e_1 + I_{rx} D_{2_y} \circ e_3 - I_{rt} I_{rx} I_{ry} \circ e_4 - I_{rt} \Delta \quad (19)$$

where

$$\Delta = \left(I_{ry}^2 D_{2_x}^2 - \frac{2}{h} I_{rx} I_{ry}^2 D_{1_x}^2 \right) \circ (u^2 + p) + \left(I_{rx}^2 D_{2_y}^2 - \frac{2}{h} I_{rx}^2 I_{ry} D_{1_y}^2 \right) \circ (v^2 + p) . \quad (20)$$

Thus, to proceed in constructing a difference Gröbner basis we have to add Δ to the set (18), and to the difference scheme e_1, e_2, e_3, e_4 too. However, in the continuous limit the difference equation $\Delta = 0$ implies the differential equation

$$u_{xx}^2 + v_{yy}^2 + p_{xx} + p_{yy} = 0 . \quad (21)$$

This follows from application of formulae (8) and (10) to (20):

$$\begin{aligned} \Delta &= 4h^2 \left(\left(u_{j+4k+2}^n - u_{j+3k+2}^n - u_{j+1k+2}^n + u_{jk+2}^n \right)^2 + \right. \\ &\quad \left. + \left(v_{j+2k+4}^n - v_{j+2k+3}^n - v_{j+2k+1}^n + v_{j+2k}^n \right)^2 + \right. \\ &\quad \left. + \left(p_{j+4k+2}^n - p_{j+3k+2}^n - p_{j+1k+2}^n + p_{jk+2}^n \right) + \right. \\ &\quad \left. \left(p_{j+2k+4}^n - p_{j+2k+3}^n - p_{j+2k+1}^n + p_{j+2k}^n \right) \right) \approx \\ &\approx 12h^4 \left(u_{j+2k+2_{xx}}^n + v_{j+2k+2_{yy}}^n + p_{j+2k+2_{xx}}^n + p_{j+2k+2_{yy}}^n + O(h^2) \right) \end{aligned}$$

The differential equation (21) does not follow from the Navier–Stokes system (1). Otherwise (21) would be an integrability condition for (1), and hence (3) could not be involutive. Therefore, the numerical integration rules used for discretization of (17), namely the trapezoidal rule for the pressure integral relations, are not consistent with the initial differential system.

6 Conclusions

By applying our algorithmic approach to generation of finite-difference schemes suggested in [2] we derived two new conservative schemes for the involutive Navier–Stokes system (1) which are similar to (although distinct from) the classical scheme derived by Harlow and Welch in [8]. However, the consistency check of the last scheme requires too cumbersome Gröbner basis computation to be done by hand whereas both our schemes admit such check done in Section 5. We have also shown that an inconsistent difference scheme can be derived by

an inappropriate choice of the numerical integration method. To check the practical quality of the new schemes one has to apply them to explicit numerical simulation of flows with appropriate initial and boundary conditions, and this is planned as our future work.

Apparently, by choosing different numerical integration methods (cf. [12]) for evaluation of the integral equations in (6) and by verification of their consistency one can generate many difference schemes for the Navier–Stokes equations. From the above described method it is also clear that it admits a natural extension to the three-dimensional Navier–Stokes equations.

Acknowledgements

The research presented in this paper was partially supported by grant 07-01-00660 from the Russian Foundation for Basic Research and by grant 1027.2008.2 from the Ministry of Education and Science of the Russian Federation.

References

1. Pozrikidis, C.: Fluid Dynamics: Theory, Computation and Numerical Simulation. Kluwer, Dordrecht (2001)
2. Gerdt, V.P., Blinkov, Y. A., Mozzhilkin, V.V.: Gröbner Bases and Generation of Difference Schemes for Partial Differential Equations. SIGMA 2, 51 (2006) arXiv:math.RA/0605334
3. Pommaret, J.F.: Partial Differential Equations and Lie Pseudogroups. Gordon & Breach, London (1978)
4. Calmet, J., Hausdorf, M., Seiler, W.M.: A Constructive Introduction to Involution. In: Akerkar, R. (ed.) Proc. Int. Symp. Applications of Computer Algebra - ISACA 2000, pp. 33–50. Allied Publishers, New Delhi (2001), <http://www.mathematik.uni-kassel.de/~seiler/>
5. Janet, M.: Leçons sur les Systèmes d'Equations aux Dérivées Partielles. Cahiers Scientifiques, IV, Gauthier-Villars, Paris (1929)
6. Gerdt, V.P., Blinkov, Y. A.: Involution Bases of Polynomial Ideals. Math. Comp. Sim. 45, 519–542 (1998) arXiv:math.AC/9912027
7. Gerdt, V.P.: Completion of Linear Differential Systems to Involution. In: Computer Algebra in Scientific Computing CASC 1999, pp. 115–137. Springer, Berlin (1999) arXiv:math.AP/9909114
8. Harlow, F.H., Welch, J.E.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys. Fluids 8(12), 2182–2189 (1965)
9. Gresho, P.M., Sani, R.L.: On Pressure Boundary Conditions for the Incompressible Navier-Stokes Equations. Int. J. Numer. Meth. Fluids 7, 1111–1145 (1987)
10. Sidorov, A.F., Shapeev, V.P., Yanenko, N.N.: Method of Differential Constraints and its Application to Gas Dynamics, Nauka, Novosibirsk (1984) (in Russian)
11. Levin, A.: Difference Algebra. Springer, Heidelberg (2008)
12. Davis, P.J., Rabinowitz, P.: Methods of Numerical Integration. Dover Publications (2007)

A Mathematica Package for Simulation of Quantum Computation

Vladimir P. Gerdt¹, Robert Kragler², and Alexander N. Prokopenya³

¹ Joint Institute for Nuclear Research
141980 Dubna, Russia
gerdt@jinr.ru

² University of Applied Sciences
D-88241 Weingarten, Germany
kragler@hs-weingarten.de

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

Abstract. In this paper we briefly describe a *Mathematica* package for simulation of quantum circuits and illustrate some of its features by simple examples. Unlike other *Mathematica*-based quantum simulators, our program provides a user-friendly graphical interface for generating quantum circuits and computing the circuit unitary matrices. It can be used for designing and testing different quantum algorithms. As an example we consider a quantum circuit implementing Grover's search algorithm and show that it gives a quadratic speed-up in solving the search problem.

1 Introduction

Quantum computation is a topic of great interest for the last two decades [1]. The main reason for this is the potential ability of a quantum computer to provide massive performance speedup in certain types of computational problems such as data searching [2], factorization [3] and encryption [4].

However, in spite of some exciting rumors that are spread by the Canadian company D-Wave (see the Web page <http://www.dwavesys.com/>), realistic quantum computers have not been built yet. Their general unavailability generates interest in developing classical simulators of quantum computation which can be used for finding and testing new efficient quantum algorithms. There is quite a number of such simulators (the corresponding links see on the website http://www.quantiki.org/wiki/index.php/List_of_QC_simulators). But most of them have been designed for solving some particular tasks such as demonstration of various quantum circuits and algorithms and, hence, can not be used for applications more general. There are also several simulators of quantum computation developed with *Mathematica* [5], for example, the *Mathematica* packages Quantum (see <http://homepage.cem.itesm.mx/lgoomez/quantum/>), Qdensity [6], QuCalc [7]. But these programs have similar shortcomings and are

not universal in a sense that it is not possible to analyze an arbitrary quantum algorithm within the framework of any of them.

Note that all quantum algorithms are traditionally expressed in the quantum circuit model [1]. And so the problem of simulating a quantum computation is reduced to construction of a quantum circuit transforming a given initial state of quantum memory register into the final state that can be measured. Such transformation is done by means of the corresponding unitary operator. Therefore, a simulator program must be able to calculate a unitary matrix corresponding to the quantum circuit in the general case of n -qubit memory register. Besides, a simulator must be a user-friendly tool which can be easily used to design and test different quantum algorithms.

We have developed the first version of a *Mathematica* package “QuantumCircuit” [8] that, in our opinion, satisfies the requirements above. At the moment we focus on constructing quantum circuits and computing the corresponding unitary transformations of quantum register but the package is improved and extended to cover all types of calculations being necessary to simulate a quantum computer. It should be noted also that in addition to the straightforward computation of the circuit matrix by means of the *Mathematica* build-in linear algebra facilities, our program provide users with the special routine to generate a system of multivariate Boolean polynomials [9] for a circuit constructed from Toffoli and Hadamard gates. This system is such that its number of common roots in the finite field \mathbb{F}_2 defines the matrix elements of the circuit matrix [10].

The paper is organized as follows. In Section 2 we briefly describe the general structure of an arbitrary quantum circuit and introduce the concept of its matrix representation. Then we describe (Section 3) the features of computing unitary matrices corresponding to different quantum gates and their sequences by applying the straightforward linear algebra procedures provided by the *Mathematica* and demonstrate examples how to compute the circuit matrix. As an application of our package in Section 4 we consider a quantum circuit implementing Grover’s search algorithm to show that repeating Grover’s iteration gives a hidden item with high probability.

2 Quantum Circuit and Its Matrix Representation

The quantum circuit model of computation is constructed analogous to the classical computing and provides an efficient and powerful language for describing quantum algorithms. A collection of n qubits forms a quantum memory register, where the input data and intermediate results of computations are held. Each qubit is a two-level quantum system that can be prepared, manipulated and measured in a controlled way. Traditionally, the state of a qubit is denoted as $|a\rangle$, corresponding to standard Dirac notation for quantum mechanical states. Therefore, a quantum memory register is shown on a diagram visualizing the circuit as a column of states of the form $|a_j\rangle$ ($j = 1, 2, \dots, n$) from which “quantum wires” start. Although a quantum circuit doesn’t contain any wires as such, the term “wires” is merely used to show evolution of qubits acted on by various quantum gates.

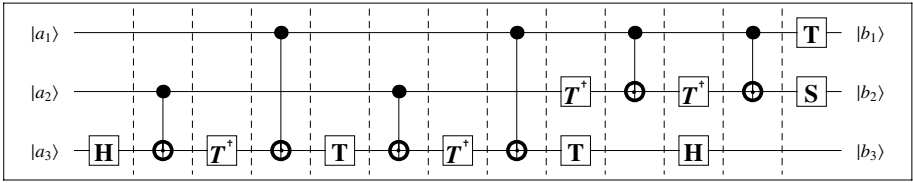


Fig. 1. Implementation of the Toffoli gate using Hadamard (H), phase (S), controlled-NOT (CNOT) and $\pi/8$ (T) gates

General structure of any quantum circuit can be readily understood from Fig. 1, where a quantum circuit implementing the Toffoli gate is depicted.

The circuit is to be read from left-to-right. It means that a column of three qubits $|a_1\rangle, |a_2\rangle, |a_3\rangle$ in the left-hand side of the diagram determines an initial state of the memory register. Then it is successively acted on by different quantum gates and its final state is shown on the right-hand side of the diagram as a column of qubits $|b_1\rangle, |b_2\rangle, |b_3\rangle$. We have drawn vertical dashed lines in Fig. 1 to show clearly that evolution of the memory register is controlled by means of successive application of quantum gates to different qubits at each step of computation.

Note that all gates are usually denoted with some symbols, for example, **H** is the Hadamard gate, “•” and symbol \oplus connected with a vertical line correspond to the control and target qubits in the controlled-NOT gate, **S** and **T** are the phase and $\pi/8$ gates, respectively, and so on (we follow here the notations of [1]). So it seems to be quite natural to introduce the following matrix for representation of the circuit shown in Fig. 1:

$$\begin{pmatrix} 1 & 1 & 1 & C & 1 & 1 & 1 & C & 1 & C & 1 & C & T \\ 1 & C & 1 & 1 & 1 & 1 & C & 1 & 1 & T^\dagger & N & T^\dagger & N & S \\ H & N & T^\dagger & N & T & N & T^\dagger & N & T & 1 & H & 1 & 1 \end{pmatrix}. \tag{1}$$

Here the unit means identical transformation of the qubit, letters “C” and “N” in the same column correspond to the control and target qubits of the controlled-NOT gate, “ T^\dagger ” denotes an adjoint gate for “T”. One can easily see that this matrix contains all the information about the structure of this circuit. It means that defining a matrix whose columns from left-to-right contain symbols corresponding to quantum gates acting on the qubits on each step of computation we can encode thus information about any quantum circuit. Obviously, number of rows in the matrix should be equal to the number of qubits in the memory register. This is an idea of our representation of a quantum circuit.

Thus, in our package a quantum circuit is represented internally as a $n \times m$ matrix, where n is a number of qubits and m determines a number of steps in the computation. It is assumed that each column contains either one multi-qubit

gate or single-qubit gates only. To decrease dimension of the matrix it is reasonable to assume also that there are no neighboring columns in the matrix containing only single-qubit gates acting on different qubits. Note that to define the corresponding matrix one can use either standard *Mathematica* representation for a matrix as list of lists or traditional notation as shown in Fig. 2. Then a function `circuit[mat]` generates a diagram corresponding to the matrix `mat`.

$$\mathbf{mat} = \left(\begin{array}{cccccccccccc} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{C} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{C} & \mathbf{1} & \mathbf{C} & \mathbf{1} & \mathbf{C} & \mathbf{T} \\ \mathbf{1} & \mathbf{C} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{C} & \mathbf{1} & \mathbf{1} & \mathbf{T}^\dagger & \mathbf{N} & \mathbf{T}^\dagger & \mathbf{N} & \mathbf{S} \\ \mathbf{H} & \mathbf{N} & \mathbf{T}^\dagger & \mathbf{N} & \mathbf{T} & \mathbf{N} & \mathbf{T}^\dagger & \mathbf{N} & \mathbf{T} & \mathbf{1} & \mathbf{H} & \mathbf{1} & \mathbf{1} \end{array} \right); \text{circuit}[\mathbf{mat}]$$

Fig. 2. A matrix corresponding to the quantum circuit of Fig. 1

It should be emphasized that a user can easily add or delete some row or column in the matrix `mat` or change some symbols replacing the corresponding quantum gates. Then running the command `circuit[mat]` immediately visualizes a new quantum circuit. Afterwards, one can easily compute a unitary matrix corresponding to the quantum circuit.

The data base of gates in our package contains the following gates [1]:

- **one-qubit gates:** Hadamard, Pauli X, Pauli Y, Pauli Z, Phase shift R_k , Phase $S \equiv R_2$ and the $\pi/8$ or $T \equiv R_3$.
- **two-qubit gates:** Controlled-X (CNOT), Controlled-Y, Controlled-Z, Controlled-S, Controlled-T, Controlled- R_k and Swap gate.
- **three-qubit gates:** Toffoli (CCNOT).

Note that each controlled gate in our program may have not only one but several control qubits and for visualization it is sufficient to write a symbol “C” in the corresponding places of the matrix `mat`. Besides, the set of available gates can be easily extended by a user.

3 Constructing the Circuit Unitary Matrix

A state of each qubit is characterized by a vector in the two-dimensional complex Hilbert space with two mutually orthogonal quantum states $|0\rangle$ and $|1\rangle$ forming a computational basis. Therefore, the state of n -qubit memory register is characterized by a vector in the 2^n -dimensional complex Hilbert space, where the basis states are given by

$$|a_1 a_2 \dots a_n\rangle \equiv |a_1\rangle \otimes |a_2\rangle \otimes \dots \otimes |a_n\rangle . \tag{2}$$

Here $a_j = 0, 1$ ($j = 1, \dots, n$) and the sign \otimes denotes tensor product of the vectors. Considering the set of digits $(a_1 a_2 \dots a_n)$ as a binary notation for the number k , the basis states (2) can be represented as column vectors:

$$\begin{aligned}
 |00\dots 00\rangle \equiv |0\rangle &= \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad |00\dots 01\rangle \equiv |1\rangle = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \dots, \\
 |11\dots 11\rangle \equiv |2^n - 1\rangle &= \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}, \tag{3}
 \end{aligned}$$

where the vector $|k\rangle$ ($k = 0, 1, \dots, 2^n - 1$) has the component $k + 1$ equal to one and all other components equal to zero.

A unitary matrix U defined by the quantum circuit with n qubits is represented as a $2^n \times 2^n$ matrix with respect to the basis states (3). As the circuit is read from left-to-right and we use the matrix **mat** to represent the circuit, the matrix U can be written as the following product

$$U = U_m U_{m-1} \dots U_2 U_1, \tag{4}$$

where U_j ($j = 1, 2, \dots, m$) is the $2^n \times 2^n$ matrix determined by the quantum gates being in the j th column of the matrix **mat**.

Remind that each column of the matrix **mat** contains either several one-qubit gates acting on different qubits or a multi-qubit gate. In the first case the corresponding matrix U_j is obtained as a tensor product of n matrices of second order representing the one-qubit gates in the computational basis. For example, the first column of the matrix (1), encoding the quantum circuit of Fig. 1, contains three symbols, namely, **1**, **1**, **H**. Therefore, the corresponding 8×8 matrix U_1 is obtained as a tensor product of the following matrices

$$\begin{aligned}
 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \\
 &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix},
 \end{aligned}$$

where we have used two 2×2 unity matrices and standard matrix representation for the Hadamard gate (see 11). Computing this tensor product can be easily implemented with built-in linear algebra *Mathematica* tools, the corresponding source code is shown in Fig. 3. However, memory resources for storing such a


```

mat1 = {{1, 0}, {0, 1}};
mat2 = {{1, 0}, {0, 1}};
mat3 =  $\frac{1}{\sqrt{2}}$  {{1, 1}, {1, -1}};
ArrayFlatten[Outer[Times,
ArrayFlatten[Outer[Times, mat1, mat2]], mat3]]

```

Fig. 3. Mathematica code implementing calculation of the matrix (4)

matrix grow exponentially with the number of qubits. And to store the $2^{10} \times 2^{10}$ unitary matrix being a tensor product of ten Hadamard gates, for example, one needs more than 1 GB memory size which limits efficient simulation of quantum circuits containing more than 10 qubits. If most of the matrix elements are zeros one can store the matrix as a sparse array and this enables to increase the number of qubits in the circuit. For example, to store the $2^{20} \times 2^{20}$ unitary matrix being a tensor product of twenty Pauli-X gates one needs only about 25 MB memory size. But in any case time of calculation grows exponentially with the number of qubits as well (see Fig. 4).

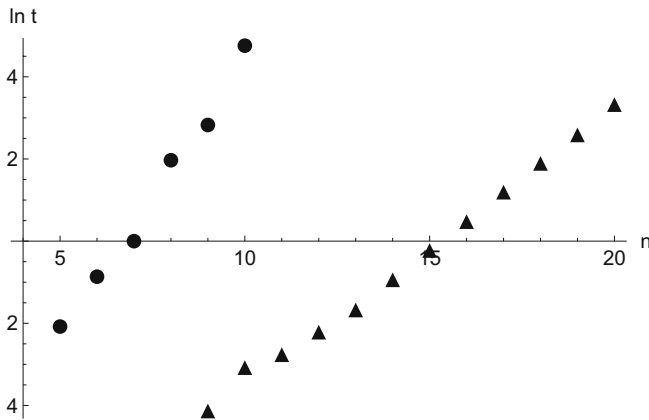


Fig. 4. Calculation time of tensor product of n Hadamard (●) and Pauli-X (▲) matrices

Note that the k th column of the matrix U_j is a vector in the 2^n -dimensional Hilbert space and is obtained as a result of the action of quantum gates, being in the j th column of the matrix **mat**, on the basis vector $|k - 1\rangle$. Therefore, if the column contains only some controlled gate, the corresponding matrix U_j may be obtained directly by specifying the transformation rules for the basis vectors (3).

For example, in the case of CNOT gate defined by the second column of the matrix (1) the Pauli-X or NOT gate is applied to the target qubit $|a_3\rangle$ only if the control qubit $|a_2\rangle$ is in the state $|1\rangle$. This operation doesn't depend on the state of the qubit $|a_1\rangle$. As a result we obtain only permutation of the basis states (3) according to the rule

$$\begin{aligned}
 |0\rangle &\equiv |000\rangle \rightarrow |000\rangle \equiv |0\rangle \\
 |1\rangle &\equiv |001\rangle \rightarrow |001\rangle \equiv |1\rangle \\
 |2\rangle &\equiv |010\rangle \rightarrow |011\rangle \equiv |3\rangle \\
 |3\rangle &\equiv |011\rangle \rightarrow |010\rangle \equiv |2\rangle \\
 |4\rangle &\equiv |100\rangle \rightarrow |100\rangle \equiv |4\rangle \\
 |5\rangle &\equiv |101\rangle \rightarrow |101\rangle \equiv |5\rangle \\
 |6\rangle &\equiv |110\rangle \rightarrow |111\rangle \equiv |7\rangle \\
 |7\rangle &\equiv |111\rangle \rightarrow |110\rangle \equiv |6\rangle
 \end{aligned} \tag{5}$$

Consequently, the columns of the matrix U_2 from left-to right are just basis vectors (3) written in the order they appear in the right-hand side of equation (5). This permutation is implemented in our package with the function `gateCN[3,3,{2}]`, the corresponding source code is shown in Fig. 5. Its arguments n , kn and kc are a number of qubits in the circuit, position of the target qubit and list of positions of the controlled qubits, respectively. The matrix U_j is stored as a sparse one what reduces the memory resources and time of calculation dramatically. For example, the computing time and memory used in the case of tensor product of eleven Hadamard gates are 10^4 times larger than similar values for the CCNOT gate in the circuit with eleven qubits.

```

gateCN[n_, kn_, kc_?ListQ] :=
Block[ {b1, u0, rules},
  b1 = Table[IntegerDigits[j, 2, n], {j, 0, 2^n - 1}];
  rules = Table[ {FromDigits[ReplacePart[b1[[j]],
    kn -> Mod[Apply[Times,
      b1[[j, kc]] ] + b1[[j, kn]], 2 ]], 2] + 1, j} -> 1,
    {j, 2^n}];
  u0 = SparseArray[rules, {2^n, 2^n}]; u0 ]
```

Fig. 5. *Mathematica* code implementing matrix representation of the CNOT gate

When all unitary matrices U_j corresponding to the columns of matrix **mat** have been calculated one can compute their product according to the expression (4) and thus find the unitary matrix of the whole circuit. Such sequence of operations in our package is done by the function `matrixU[mat]` which is called with a single argument that is just the matrix **mat** encoding the circuit. For the matrix (1), for example, the corresponding command gives the unitary 8×8 matrix shown in Fig. 6. It should be noted that this unitary matrix coincides with the matrix given by the function `gateCN[3, 3, {1,2}]` which generates the

unitary matrix corresponding to the Toffoli gate. Therefore, one can conclude that the circuit shown in Fig. 6 implements the Toffoli gate. This is a direct proof of the statement that Toffoli gate can be composed from the Hadamard, phase, CNOT and $\pi/8$ gates. This example demonstrates one application of our package as a tool for proving the equivalence of different quantum circuits.

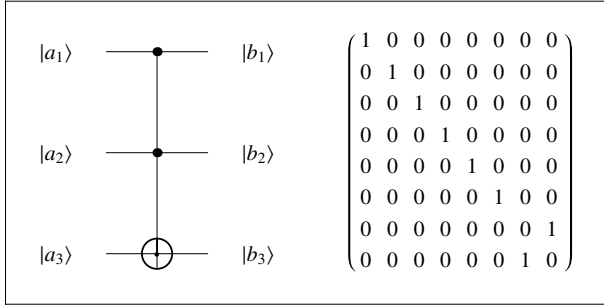


Fig. 6. Toffoli gate and its unitary matrix

It should be emphasized that a matrix **mat** representing some quantum circuit may be not only constructed by hand as it has been done above but may be generated by some function as well. For example, to generate a matrix modeling a quantum circuit implementing the Fourier transform for n -qubit memory register one can define the function **modelFourier[n]** (Fig. 8). Then using the functions **circuit** and **matrixU**, we can easily visualize the circuit (Fig. 7) and compute the corresponding unitary matrix.

```

modelFourier [n_] := Module [{model, mm, n1},
  model = Array [mm, {n, n (n + 1) / 2}] /. mm [i_, j_] → 1;
  n1 = 0;
  Do [ Do [ If [k == 1, model [[j, n1 + 1]] = H,
    (model [[j, n1 + k]] = C; model [[j + k - 1, n1 + k]] = Rk)
    ], {k, n - j + 1}];
  n1 = n1 + n - j + 1, {j, n}; model ]

```

Fig. 7. Mathematica code for generation of a matrix representing the circuit for the quantum Fourier transform

Note that a unitary matrix corresponding to the quantum Fourier transform may be also computed with the Mathematica package Qdensity [6] but this is done by a procedure written specially for this quantum circuit. However, time of calculation is only a little bit less than in the case of our package “QuantumCircuit”, whereas the function **matrixU** is universal, and of course this time grows exponentially with the number of qubits for both programs (see Fig. 9).

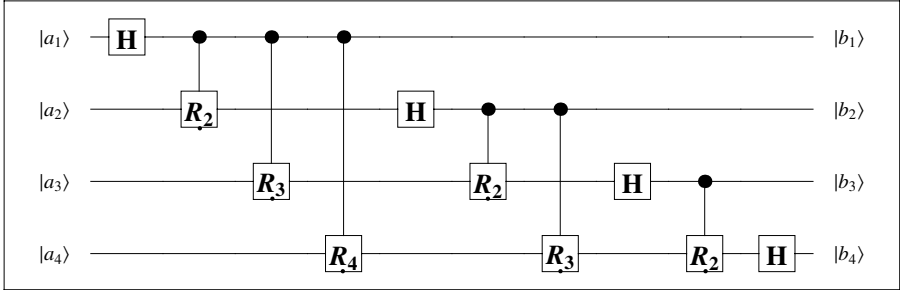


Fig. 8. A quantum circuit implementing the Fourier transform ($n = 4$)

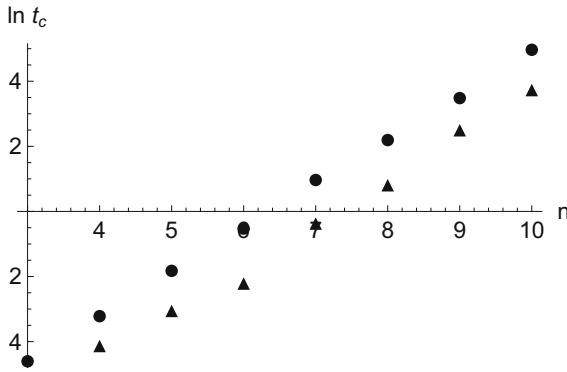


Fig. 9. Time of computing a unitary matrix for the Fourier transform with our package “QuantumCircuit” (●) and Qdensity (▲)

4 Implementation of the Grover Search Algorithm

To demonstrate application of the package “QuantumCircuit” for analysis of quantum circuits let us consider Grover’s search algorithm [2] as an example. The search problem is formulated as follows [11]: some n -bit integer k is hidden in a black-boxed subroutine that indicates, when presented with any n -bit integer x , whether or not x coincides with k , returning this information as the value of the n -bit binary function. The problem is to find k with minimum number of applying the subroutine. Let a quantum memory register contain five qubits ($n = 4$): four qubits $|a_1\rangle, |a_2\rangle, |a_3\rangle, |a_4\rangle$ are originally prepared in the state $|0\rangle$ and one ancillary qubit $|a_5\rangle$ is in the state $|1\rangle$. It means that the initial state of the memory register is $|00001\rangle$ and the corresponding basis vector in the 32-dimensional Hilbert space has the second component equal to one and all other components equal to zero (see equation (3)). Applying five Hadamard gates, one for each qubit, we obtain an equal superposition of all basis states of the five-qubit system. A quantum search subroutine bounded by two dashed lines

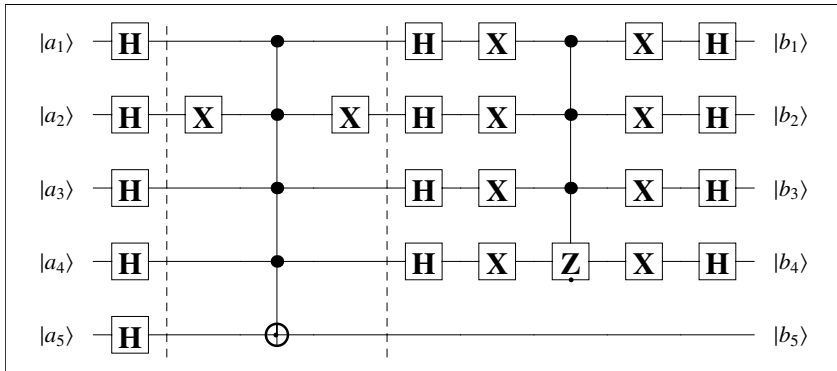


Fig. 10. Quantum circuit implementing Grover’s search algorithm ($n = 4, k = 11$)

in the diagram (Fig. 10) is a 4-bit binary function that outputs 1 if its input is some given integer ($k = 11$ in the case shown) and 0 otherwise. This subroutine together with a quantum circuit drawn on the right of the dashed line form one Grover’s iteration.

Note that Grover’s iteration can be applied to the memory register several times bringing it to some final state that can be measured in the computational basis. And if the number of iterations is equal to the integer part of $\pi/(4 \arcsin(2^{-n/2})) = \pi/(4 \arcsin(1/4)) = 3.108$ (see [1], [2], [11]), then the final state will be exactly $|k\rangle$ with very high probability.

To find a final state of the memory register after several Grover’s iterations let us define two matrices **mat0** and **matG** (see Fig. 11). The first one represents a column of Hadamard gates bringing initial state $|00001\rangle$ of the memory register to equal superposition of all basis states. The corresponding unitary $2^5 \times 2^5$ matrix **matU0** is given by the function **matrixU[mat0]**.

```

initial = SparseArray[2 -> 1, 2^5];
mat0 = {{H}, {H}, {H}, {H}, {H}}; matU0 = matrixU[mat0];
matG =  $\begin{pmatrix} 1 & C & 1 & H & X & C & X & H \\ X & C & X & H & X & C & X & H \\ 1 & C & 1 & H & X & C & X & H \\ 1 & C & 1 & H & X & Z & X & H \\ 1 & N & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$ ; matU1 = matrixU[matG];

firstIteration = matU1.matU0.initial;
secondIteration = matU1.firstIteration;
thirdIteration = matU1.secondIteration

```

Fig. 11. Mathematica code for computing a final vector after three Grover’s iterations

The second matrix **matG** represents one Grover’s iteration and the corresponding unitary matrix is given by **matrixU[matG]**. Defining the initial state of the memory register as a sparse vector **initial**, one can act on it with operator **matU0** and then apply successively several Grover’s iterations **matU1**. As a result we obtain a unit vector with $2^5 = 32$ components which determine probabilities of different basis states of the memory register. Remind that hidden item is encoded by the states of four qubits $|a_1\rangle, |a_2\rangle, |a_3\rangle, |a_4\rangle$, while the ancillary qubit $|a_5\rangle$ is finally in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ and may be found in both basis states $|0\rangle$ and $|1\rangle$ with equal probability. Therefore, a probability P to get k ($k = 0, 1, \dots, 15$) as a result of measurement of the memory register $|a_1a_2a_3a_4\rangle$ is equal to a sum of the $2k$ th and $(2k + 1)$ th components squared of the final vector.

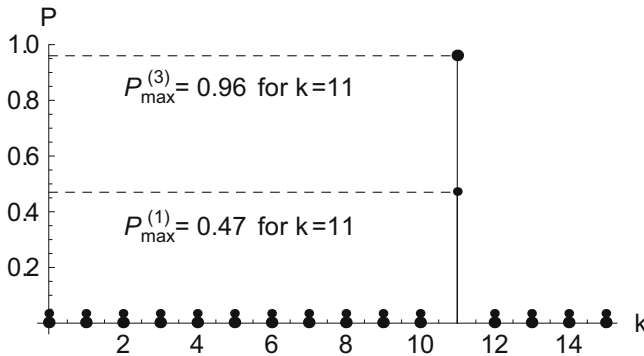


Fig. 12. Probability distribution in the final state after one and three Grover’s iterations

Fig. 12 shows that after one iteration a probability to get a correct number $k = 11$ as a result of measurement is equal to 47 percent, while after the third Grover’s iteration a standard measurement in the computational basis gives 11 with probability 96 percent. It should be noted also that the fourth iteration decreases a probability to get a correct result to 58 percent. Thus, maximum probability to obtain correct result is reached if the number of iterations is equal to its optimal value that is determined as an integer part of $\pi/(4 \arcsin(1/\sqrt{N}))$, where $N = 2^n$. For large values of N this number is $O(\sqrt{N})$ and, hence, Grover’s algorithm provides a quadratic speed-up in solving the search problem in comparison with a classical computer which requires $O(N)$ applications of the subroutine.

5 Conclusion

In this paper we present a *Mathematica* package for simulation of quantum circuits we are currently working on. The package provides a user-friendly graphical interface for generating quantum circuits and computing the circuit unitary

matrices. Arbitrary circuit is represented internally as a symbolic table whose elements correspond to different one- and multi-qubit gates. Its unitary matrix is computed by means of the *Mathematica* build-in linear algebra facilities.

Estimating the time of calculation of the unitary matrix corresponding to the quantum Fourier transform with our package and with the *Mathematica* package Qdensity [6] shows that they are comparable (see Fig. 9). But the package Qdensity implements the standard algorithm for computing this matrix what means that simulating other quantum circuit requires to design new algorithm of calculation. In our package the corresponding function `matrixU` for computing the unitary matrix is quite universal. Therefore, our package “QuantumCircuit” is universal and can be used for designing and testing different quantum algorithms.

Acknowledgements

The contribution of one of the authors (V.P.G.) research was partially supported by grant 07-01-00660 from the Russian Foundation for Basic Research.

References

1. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
2. Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett. 79, 325–328 (1997)
3. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comp. 26(5), 1484–1509 (1997)
4. Phoenix, S.J.D., Townsend, P.D.: Quantum cryptography: how to beat the code breakers using quantum mechanics. Contemp. Phys. 36, 165–195 (1995)
5. Wolfram, S.: The Mathematica Book, 4th edn. Wolfram Media/Cambridge University Press (1999)
6. Julia-Diaz, B., Burdis, J.M., Tabakin, F.: QDENSITY – A Mathematica quantum computer simulation. Computer Physics Comm. 174(11), 914–934 (2006)
7. Phillips, F.: Editor’s Pick: Quantum computation. The Mathematica Journal 8(1) (2001)
8. Gerdt, V.P., Kragler, R., Prokopenya, A.N.: A Mathematica Package for Construction of Circuit Matrices in Quantum Computation. In: Computer Algebra and Differential Equations. Acta Academiae Aboensis, Ser. B, vol. 67(2), pp. 28–38 (2007)
9. Rudeanu, S.: Boolean functions and equations. North-Holland Publishing Co., Amsterdam. American Elsevier Publishing Co., Inc., New York (1974)
10. Dawson, C.M., Haselgrove, H.L., Hines, A.P., et al.: Quantum computing and polynomial equations over the finite field Z_2 . Quantum Information and Computation 5(2), 102–112 (2005) arXiv:quant-ph/0408129
11. Mermin, N.D.: Quantum computer science. An introduction. Cambridge University Press, Cambridge (2007)

On Computing the Hermite Form of a Matrix of Differential Polynomials

Mark Giesbrecht and Myung Sub Kim

Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

Abstract. Given a matrix $A \in F(t)[\mathcal{D}; \delta]^{n \times n}$ over the ring of differential polynomials, we show how to compute the Hermite form H of A and a unimodular matrix U such that $UA = H$. The algorithm requires a polynomial number of operations in F in terms of n , $\deg_{\mathcal{D}} A$, $\deg_t A$. When $F = \mathbb{Q}$ it require time polynomial in the bit-length of the rational coefficients as well.

1 Introduction

Canonical forms of matrices over principal ideal domains (such as \mathbb{Z} or $F[x]$, for a field F) have proven invaluable for both mathematical and computational purposes. One of the successes of computer algebra over the past three decades has been the development of fast algorithms for computing these canonical forms. These include triangular forms such as the Hermite form (Hermite, 1863), low degree forms like the Popov form (Popov, 1972), as well as the diagonal Smith form (Smith, 1861).

Canonical forms of matrices over non-commutative domains, especially rings of differential and difference operators, are also extremely useful. These have been examined at least since Dickson (1923), Wedderburn (1932), and Jacobson (1943). A typical domain under consideration is that of differential polynomials. For our purposes these are polynomials over a function field $F(t)$ (where F is a field of characteristic zero, typically an extension of \mathbb{Q} , or some representation of \mathbb{C}). A differential indeterminate \mathcal{D} is adjoined to form the *ring of differential polynomials* $F(t)[\mathcal{D}; \delta]$, which consists of the polynomials in $F(t)[\mathcal{D}]$ under the usual addition and a non-commutative multiplication defined such that $\mathcal{D}a = a\mathcal{D} + \delta(a)$, for any $a \in F(t)$. Here $\delta : F(t) \rightarrow F(t)$ is a *pseudo-derivative*, a function such that for all $a, b \in F(t)$ we have

$$\delta(a + b) = \delta(a) + \delta(b) \quad \text{and} \quad \delta(ab) = a\delta(b) + \delta(a)b.$$

The most common derivation in $F(t)$ takes $\delta(a) = a'$ for any $a \in F(t)$, the usual derivative of a , though other derivations (say $\delta(t) = t$) are certainly of interest.

A primary motivation in the definition of $F(t)[\mathcal{D}; \delta]$ is that there is a natural action on the space of infinitely differentiable functions in t , namely the differential polynomial

$$a_m \mathcal{D}^m + a_{m-1} \mathcal{D}^{m-1} + \cdots + a_1 \mathcal{D} + a_0 \in F(t)[\mathcal{D}; \delta]$$

acts as the linear differential operator

$$a_m(t) \frac{d^m y(t)}{dt^m} + a_{m-1}(t) \frac{d^{m-1} y(t)}{dt^{m-1}} + \cdots + a_1(t) \frac{dy(t)}{dt} + a_0(t) y(t)$$

on a differentiable function $y(t)$. Solving and analyzing systems of such operators involves working with matrices over $F(t)[\mathcal{D}; \delta]$, and invariants such as the differential analogues of the Smith, Popov and Hermite forms provide important structural information.

In commutative domains such as \mathbb{Z} and $F[x]$, it has been more common to compute the triangular Hermite and diagonal Smith form (as well as the lower degree Popov form, especially as an intermediate computation). Indeed, these forms are more canonical in the sense of being canonical in their class under multiplication by unimodular matrices. Polynomial-time algorithms for the Smith and Hermite forms over $F[x]$ were developed by Kannan (1985), with important advances by Kaltofen et al. (1987), Villard (1995), Mulders and Storjohann (2003), and many others. One of the key features of this recent work in computing normal forms has been a careful analysis of the complexity in terms of matrix size, entry degree, and coefficient swell. Clearly identifying and analyzing the cost in terms of all these parameters has led to a dramatic drop in both theoretical and practical complexity.

Computing the classical Smith and Hermite forms of matrices over differential (and more general Ore) domains has received less attention though normal forms of differential polynomial matrices have applications in solving differential systems and control theory. Abramov and Bronstein (2001) analyzes the number of reduction steps necessary to compute a row-reduced form, while Beckermann et al. (2006) analyze the complexity of row reduction in terms of matrix size, degree and the sizes of the coefficients of some shifts of the input matrix. Beckermann et al. (2006) demonstrates tight bounds on the degree and coefficient sizes of the output, which we will employ here. For the Popov form, Cheng (2003) gives an algorithm for matrices of shift polynomials. Cheng's approach involves order bases computation in order to eliminate lower order terms of Ore polynomial matrices. A main contribution of Cheng (2003) is to give an algorithm computing the row rank and a row-reduced basis of the left nullspace of a matrix of Ore polynomials in a fraction-free way. This idea is extended in Davies et al. (2008) to compute Popov form of general Ore polynomial matrices. In Davies et al. (2008), they reduce the problem of computing Popov form to a nullspace computation. However, though Popov form is useful for rewriting high order terms with respect to low order terms, we want a different normal form more suited to solving system of linear diophantine equations. Since the Hermite form is upper triangular it meets this goal nicely, not to mention the fact that it is a "classical" canonical form. In a slightly different vein, Middeke (2008) has recently given an algorithm for the Smith (diagonal) form of a matrix of differential polynomials, which requires time polynomial in the matrix size and degree (but the coefficient size is not analyzed).

In this paper, we first discuss some basic operations with polynomials in $F(t)[\mathcal{D}; \delta]$, which are typically written with respect to the differential variable \mathcal{D} as

$$f = f_0 + f_1\mathcal{D} + f_2\mathcal{D}^2 + \cdots + f_d\mathcal{D}^d, \tag{1.1}$$

where $f_0, \dots, f_d \in F(t)$, with $f_d \neq 0$. We write $d = \deg_{\mathcal{D}} f$ to mean the degree in the differential variable, and generally refer to this as the *degree* of f . Since this is a non-commutative ring, it is important to set a standard notation in which the coefficients $f_0, \dots, f_d \in F(t)$ are written to the left of the differential variable \mathcal{D} . For $u, v \in F[t]$ relatively prime, we can define $\deg_t(u/v) = \max\{\deg_t u, \deg_t v\}$. This is extended to $f \in F(t)[\mathcal{D}; \delta]$ as in (1.1) by letting $\deg_t f = \max_i\{\deg_t f_i\}$. We think of \deg_t as measuring coefficient size or height. Indeed, with a little extra work the bounds and algorithms in this paper are effective over $\mathbb{Q}(t)$ as well, where we also include the bit-length of rational coefficients, as well as the degree in t , in our analyses.

A matrix $U \in F(t)[\mathcal{D}; \delta]^{n \times n}$ is said to be *unimodular* if there exists a $V \in F(t)[\mathcal{D}; \delta]^{n \times n}$ such that $UV = I$, the $n \times n$ identity matrix. Note that we do not employ the typical determinantal definition of a unimodular matrix, as there is no easy notion of determinant for matrices over $F(t)[\mathcal{D}; \delta]$ (indeed, working around this deficiency suffices much of our work).

A matrix $H \in F(t)[\mathcal{D}; \delta]^{n \times n}$ is said to be in *Hermite form* if H is upper triangular, if every diagonal entry is monic, and every off-diagonal entry has degree less than the diagonal entry below it. As an example, the matrix

$$\begin{pmatrix} 1 + (t + 2)\mathcal{D} + \mathcal{D}^2 & 2 + (2t + 1)\mathcal{D} & 1 + (1 + t)\mathcal{D} \\ 2t + t^2 + t\mathcal{D} & 2 + 2t + 2t^2 + \mathcal{D} & 4t + t^2 \\ 3 + t + (3 + t)\mathcal{D} + \mathcal{D}^2 & 8 + 4t + (5 + 3t)\mathcal{D} + \mathcal{D}^2 & 7 + 8t + (2 + 4t)\mathcal{D} \end{pmatrix}$$

has Hermite form

$$\begin{pmatrix} 2 + t + \mathcal{D} & 1 + 2t & \frac{-2+t+2t^2}{2t} - \frac{1}{2t}\mathcal{D} \\ 0 & 2 + t + \mathcal{D} & 1 + \frac{7t}{2} + \frac{1}{2}\mathcal{D} \\ 0 & 0 & -\frac{2}{t} + \frac{-1+2t+t^2}{t}\mathcal{D} + \mathcal{D}^2 \end{pmatrix}.$$

Note that the Hermite form may have denominators in t . Also, while this example does not demonstrate it, it is common that the degrees in the Hermite form, in both t and \mathcal{D} , are substantially larger than in the input.

In this paper we will only concern ourselves with matrices in $F(t)[\mathcal{D}; \delta]^{n \times n}$ of full row rank, that is, matrices whose rows are $F(t)[\mathcal{D}; \delta]$ -linear independent. For any matrix $A \in F(t)[\mathcal{D}; \delta]^{n \times n}$, we show there exists a unimodular matrix U such that $UA = H$ is in Hermite form. This form is canonical in the sense that if two matrices $A, B \in F(t)[\mathcal{D}; \delta]^{n \times n}$ are such that $A = PB$ for unimodular $P \in F(t)[\mathcal{D}; \delta]^{n \times n}$ then the Hermite form of A equals the Hermite form of B .

The main contribution of this paper is an algorithm that, given a matrix $A \in F(t)[\mathcal{D}; \delta]^{n \times n}$ (of full row rank), computes H and U such that $UA = H$, which requires a polynomial number of F-operations in n , $\deg_{\mathcal{D}} A$, and $\deg_t A$. It will also require time polynomial in the coefficient bit-length when $F = \mathbb{Q}$.

The remainder of the paper is organized as follows. In Section 2 we summarize some basic properties of differential polynomial rings and present and analyze algorithms for some necessary basic operations. In Section 3 we introduce a new approach to compute appropriate degree bounds on the coefficients of H and U . In Section 4 we present our algorithm for computing the Hermite form of a matrix of differential polynomials and analyze it completely.

2 Basic Structure and Operations in $F[t][\mathcal{D}; \delta]$

In this section we discuss some of the basic structure of the ring $F(t)[\mathcal{D}; \delta]$ and present and analyze simple algorithms to do some computations that will be necessary in the next section.

Some well-known properties of $F(t)[\mathcal{D}; \delta]$ are worth recalling; see [Bronstein and Petkovšek \(1994\)](#) for an algorithmic presentation of this theory. Given $f, g \in F(t)[\mathcal{D}; \delta]$, there is a degree function (in \mathcal{D}) which satisfies the usual properties: $\deg_{\mathcal{D}}(fg) = \deg_{\mathcal{D}}f + \deg_{\mathcal{D}}g$ and $\deg_{\mathcal{D}}(f + g) \leq \max\{\deg_{\mathcal{D}}f, \deg_{\mathcal{D}}g\}$. $F(t)[\mathcal{D}; \delta]$ is also a left and right principal ideal ring, which implies the existence of a right (and left) division with remainder algorithm such that there exists unique $q, r \in F(t)[\mathcal{D}; \delta]$ such that $f = qg + r$ where $\deg_{\mathcal{D}}(r) < \deg_{\mathcal{D}}(g)$. This allows for a right (and left) euclidean-like algorithm which shows the existence of a greatest common right divisor, $h = \text{gcd}(f, g)$, a polynomial of minimal degree (in \mathcal{D}) such that $f = uh$ and $g = vh$ for $u, v \in F(t)[\mathcal{D}; \delta]$. The GCRD is unique up to a left multiple in $F(t) \setminus \{0\}$, and there exist co-factors $a, b \in F(t)[\mathcal{D}; \delta]$ such that $af + bg = \text{gcd}(f, g)$. There also exists a least common left multiple $\text{lclm}(f, g)$. Analogously there exists a greatest common left divisor, $\text{gcdl}(f, g)$, and least common right multiple, $\text{lcrm}(f, g)$, both of which are unique up to a right multiple in $F(t)$.

Efficient algorithms for computing products of polynomials are developed in [van der Hoeven \(2002\)](#) and [Bostan et al. \(2008\)](#), while fast algorithms to compute the LCLM and GCRD, are developed in [Li and Nemes \(1997\)](#) and [Li \(1998\)](#). In this paper we will only need to compute very specific products of the form $\mathcal{D}^k f$ for some $k \in \mathbb{N}$. We will work with differential polynomials in $F[t][\mathcal{D}; \delta]$, as opposed to $F(t)[\mathcal{D}; \delta]$, and manage denominators separately. If $f \in F[t][\mathcal{D}; \delta]$ is written as in [\(1.1\)](#), then $f_0, \dots, f_d \in F[t]$, and

$$\mathcal{D}f = \sum_{0 \leq i \leq d} f_i \mathcal{D}^{i+1} + \sum_{0 \leq i \leq d} f'_i \mathcal{D}^i \in F[t][\mathcal{D}; \delta],$$

where $f'_i \in F[t]$ is the usual derivative of $f_i \in F[t]$. Assume $\deg_t f \leq e$. It is easily seen that $\deg_{\mathcal{D}}(\mathcal{D}f) = d + 1$, and $\deg_t(\mathcal{D}f) \leq e$. The cost of computing $\mathcal{D}f$ is $O(de)$ operations in F . Computing $\mathcal{D}^k f$, for $1 \leq k \leq m$ then requires $O(dem)$ operations in F .

If $F = \mathbb{Q}$ we must account for the bit-length of the coefficients as well. Assuming our polynomials are in $\mathbb{Z}[t][\mathcal{D}; \delta]$ (which will be sufficient), and are written as above, we have $f_i = \sum_{0 \leq j \leq e} f_{ij} t^j$ for $f_{ij} \in \mathbb{Z}$. We write $\|f\|_{\infty} = \max |f_{ij}|$ to capture the coefficient size of f . It easily follows that $\|\mathcal{D}f\|_{\infty} \leq (e + 1)\|f\|_{\infty}$, and so $\|\mathcal{D}^m f\|_{\infty} \leq (e + 1)^m \|f\|_{\infty}$.

Lemma 2.1

- (i) Let $f \in \mathbb{F}[t][\mathcal{D}; \delta]$ have $\deg_{\mathcal{D}} f = d$, $\deg_t f = e$, and let $m \in \mathbb{N}$. Then we can compute $\mathcal{D}^k f$, for $1 \leq k \leq m$, with $O(\text{dem})$ operations in \mathbb{F} .
- (ii) Let $f \in \mathbb{Z}[t][\mathcal{D}; \delta]$. Then $\|\mathcal{D}^m f\|_{\infty} \leq (e + 1)^m \cdot \|f\|_{\infty}$, and we can compute $\mathcal{D}^i f$, for $1 \leq i \leq m$, with $O(\text{dem} \cdot (m \log e + \log \|f\|_{\infty})^2)$ bit operations.

We make no claim that the above methods are the most efficient, and faster polynomial and matrix arithmetic will certainly improve the cost. However, the above analysis will be sufficient, and these costs will be dominated by others in the algorithms of later sections.

3 Existence and Degree Bounds on the Hermite Form

In this section we prove the existence and uniqueness of the Hermite form over $\mathbb{F}(t)[\mathcal{D}; \delta]$, and prove some important properties about unimodular matrices and equivalence over this ring. The principal technical difficulty is that there is no natural determinant function with the properties found in commutative linear algebra. The determinant is one of the main tools used in the analysis of essentially all fast algorithms for computing the Hermite form H and transformation matrix U , and specifically two relevant techniques in established methods by [Storjohann \(1994\)](#) and [Kaltofen et al. \(1987\)](#). One approach might be to employ the non-commutative determinant of [Dieudonné \(1943\)](#), but this adds considerable complication. Instead, we find degree bounds via established bounds on the row-reduced form.

Definition 3.1 (Unimodular matrix). Let $U \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$ and suppose there exists a $V \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$ such that $UV = I_n$, where I_n is the identity matrix over $\mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$. Then U is called a unimodular matrix over $\mathbb{F}(t)[\mathcal{D}; \delta]$.

This definition is in fact symmetric, in that V is also unimodular, as shown in the following lemma (the proof of which is left to the reader).

Lemma 3.1. Let $U \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$ be unimodular such that there exists a $V \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$ with $UV = I_n$. Then $VU = I_n$ as well.

Theorem 3.1. Let $a, b \in \mathbb{F}(t)[\mathcal{D}; \delta]$. There exists a unimodular matrix

$$W = \begin{pmatrix} u & v \\ s & t \end{pmatrix} \in \mathbb{F}(t)[\mathcal{D}; \delta]^{2 \times 2} \text{ such that } W \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} g \\ 0 \end{pmatrix},$$

where $g = \text{gcd}(a, b)$ and $sa = -tb = \text{lcm}(a, b)$.

Proof. Let $u, v \in \mathbb{F}(t)[\mathcal{D}; \delta]$ be the multipliers from the euclidean algorithm such that $ua + vb = g$. Since $sa = -tb = \text{lcm}(a, b)$, we know that $\text{gcd}(s, t) = 1$ (otherwise the minimality of the degree of the lcm would be violated). It follows that there exist $c, d \in \mathbb{F}(t)[\mathcal{D}; \delta]$ such that $sc + td = 1$. Now observe that

$$\begin{pmatrix} u & v \\ s & t \end{pmatrix} \begin{pmatrix} ag^{-1} & c \\ bg^{-1} & d \end{pmatrix} \begin{pmatrix} 1 - uc - vd \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & uc + vd \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 - uc - vd \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Thus

$$W^{-1} = \begin{pmatrix} ag^{-1} & ag^{-1}(-uc - vd) + c \\ bg^{-1} & bg^{-1}(-uc - vd) + d \end{pmatrix} = \begin{pmatrix} ag^{-1} & -a + c \\ bg^{-1} & -b + d \end{pmatrix},$$

so W is unimodular. □

Definition 3.2 (Hermite Normal Form). *Let $H \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$ with full row rank. The matrix H is in Hermite form if H is upper triangular, if every diagonal entry of H is monic, and if every off-diagonal entry of H has degree (in \mathcal{D}) strictly lower than the degree of the diagonal entry below it.*

Theorem 3.2. *Let $A \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$ have row rank n . Then there exists a matrix $H \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$ with row rank n in Hermite form, and a unimodular matrix $U \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$, such that $UA = H$.*

Proof. We show this induction on n . The base case, $n = 1$, is trivial and we suppose that the theorem holds for $(n - 1) \times (n - 1)$ matrices. Since A has row rank n , we can find a permutation of the rows of A such that every principal minor of A has full row rank. Since this permutation is a unimodular transformation of A , we assume this property about A . Thus, by the induction hypothesis, there exists a unimodular matrix $U_1 \in \mathbb{F}(t)[\mathcal{D}; \delta]^{(n-1) \times (n-1)}$ such that

$$\begin{pmatrix} & 0 \\ U_1 & 0 \\ & \vdots \\ & 0 \\ 0 & 0 \cdots 0 & 1 \end{pmatrix} \cdot A = \bar{H} = \begin{pmatrix} \bar{H}_{1,1} & \cdots & \cdots & * & * \\ & \bar{H}_{2,2} & \cdots & * & * \\ & & \ddots & \vdots & \vdots \\ 0 & & & \bar{H}_{n-1,n-1} & * \\ A_{n,1} & A_{n,2} & \cdots & A_{n,n-1} & A_{n,n} \end{pmatrix} \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n},$$

where the $(n - 1)$ st principal minor of \bar{H} is in Hermite form. By Theorem 3.1, we know that there exists a unimodular matrix

$$W = \begin{pmatrix} u_i & v_i \\ s_i & -t_i \end{pmatrix} \in \mathbb{F}(t)[\mathcal{D}; \delta]^{2 \times 2} \text{ such that } W \begin{pmatrix} \bar{H}_{ii} \\ A_{n,i} \end{pmatrix} = \begin{pmatrix} g_i \\ 0 \end{pmatrix} \in \mathbb{F}(t)[\mathcal{D}; \delta]^{2 \times 1}.$$

This allows us to reduce $A_{n,1}, \dots, A_{n,n-1}$ to zero, and does not introduce any non-zero entries below the diagonal. Also, all off-diagonal entries can be reduced using unimodular operations modulo the diagonal entry, putting the matrix into Hermite form. □

Corollary 3.1. *Let $A \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$ have full row rank. Suppose $UA = H$ for unimodular $U \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$ and Hermite form $H \in \mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$. Then both U and H are unique.*

Proof. Suppose H and G are both Hermite forms of A . Thus, there exist unimodular matrices U and V such that $UA = H$ and $VA = G$, and $G = WH$ where $W = VU^{-1}$ is unimodular. Since G and H are upper triangular matrices, we know W is as well. Moreover, since G and H have monic diagonal entries, the diagonal entries of W equal 1. We now prove W is the identity matrix. By

way of contradiction, first assume that W is not the identity, so there exists an entry W_{ij} which is the first nonzero off-diagonal entry on the i th row of W . Since $i < j$ and since $W_{ii} = 1$, $G_{ij} = H_{ij} + W_{ij}H_{jj}$. Because $W_{ij} \neq 0$, we see $\deg_{\mathcal{D}}G_{ij} \geq \deg_{\mathcal{D}}G_{jj}$, which contradicts the definition of the Hermite form. The uniqueness of U follows similarly. \square

Definition 3.3 (Row Degree). A matrix $T \in F(t)[\mathcal{D}; \delta]^{n \times n}$ has row degree $\vec{u} \in (\mathbb{N} \cup \{-\infty\})^n$ if the i th row of T has degree u_i . We write $\text{rowdeg } \vec{u}$.

Definition 3.4 (Leading Row Coefficient Matrix). Let $T \in F(t)[\mathcal{D}; \delta]^{n \times n}$ have $\text{rowdeg } \vec{u}$. Set $N = \deg_{\mathcal{D}}T$ and $S = \text{diag}(\mathcal{D}^{N-u_1}, \dots, \mathcal{D}^{N-u_n})$. We write

$$ST = LD^N + \text{lower degree terms in } \mathcal{D},$$

where the matrix $L = LC_{\text{row}}(T) \in F(t)^{n \times n}$ is called the leading row coefficient matrix of T .

Definition 3.5 (Row-reduced Form). A matrix $T \in F(t)[\mathcal{D}; \delta]^{m \times s}$ with rank r is in row-reduced form if $\text{rank } LC_{\text{row}}(T) = r$.

Fact 3.1 (Beckermann et al. (2006) Theorem 2.2). For any $A \in F(t)[\mathcal{D}; \delta]^{m \times s}$ there exists a unimodular matrix $U \in F(t)[\mathcal{D}; \delta]^{m \times m}$, with $T = UA$ having $r \leq \min\{m, s\}$ nonzero rows, $\text{rowdeg}T \leq \text{rowdeg}A$, and where the submatrix consisting of the r nonzero rows of T are row-reduced. Moreover, the unimodular multiplier satisfies the degree bound

$$\text{rowdeg}U \leq \vec{v} + (|\vec{u}| - |\vec{v}| - \min_j \{u_j\}) \vec{e},$$

where $\vec{u} := \max(\vec{0}, \text{rowdeg}A)$, $\vec{v} := \max(\vec{0}, \text{rowdeg}T)$, and \vec{e} is the column vector with all entries equal to 1.

The proof of the following is left to the reader.

Corollary 3.2. If $A \in F(t)[\mathcal{D}; \delta]^{n \times n}$ is a unimodular matrix then the row reduced form of A is an identity matrix.

The following theorems provide degree bounds on H and U . We first compute a degree bound of the inverse of U by using the idea of backward substitution, and then use the result of Beckermann et al. (2006) to compute degree bound of U .

Theorem 3.3. Let $A \in F(t)[\mathcal{D}; \delta]^{n \times n}$ be a matrix with $\deg_{\mathcal{D}}A_{ij} \leq d$ and full row rank. Suppose $UA = H$ for unimodular matrix $U \in F(t)[\mathcal{D}; \delta]^{n \times n}$ and $H \in F(t)[\mathcal{D}; \delta]^{n \times n}$ in Hermite form. Then there exist a unimodular matrix $V \in F(t)[\mathcal{D}; \delta]^{n \times n}$ such that $A = VH$ where $UV = I_n$ and $\deg_{\mathcal{D}}V_{ij} \leq d$.

Proof. We prove by induction on n . The base case is $n = 1$. Since $H_{11} = \text{gcd}(A_{11}, \dots, A_{n1})$, $\deg_{\mathcal{D}}H_{11} \leq d$ and so $\deg_{\mathcal{D}}V_{i1} \leq d$ for $1 \leq i \leq n$. Now,

we suppose that our claim is true for k where $1 < k < n$. Then we have to show that $\deg_{\mathcal{D}}V_{ik+1} \leq d$. We need to consider two cases:

Case 1: $\deg_{\mathcal{D}}V_{i,k+1} > \max(\deg_{\mathcal{D}}V_{i1}, \dots, \deg_{\mathcal{D}}V_{ik})$. Since

$$\begin{aligned} \deg_{\mathcal{D}}H_{k+1,k+1} &\geq \max(\deg_{\mathcal{D}}H_{1,k+1}, \dots, \deg_{\mathcal{D}}H_{k,k+1}), \\ \deg_{\mathcal{D}}A_{i,k+1} &= \deg_{\mathcal{D}}(V_{i,k+1}H_{k+1,k+1}), \end{aligned}$$

where $A_{i,k+1} = V_{i1}H_{1,k+1} + \dots + V_{i,k+1}H_{k+1,k+1}$. Thus, $\deg_{\mathcal{D}}V_{i,k+1} \leq d$.

Case 2: $\deg_{\mathcal{D}}V_{i,k+1} \leq \max(\deg_{\mathcal{D}}V_{i1}, \dots, \deg_{\mathcal{D}}V_{ik})$. Thus, by induction hypothesis, $\deg_{\mathcal{D}}V_{i,k+1} \leq d$. □

Corollary 3.3. *Let A , V , and U be those in Theorem 3.3. Then $\deg_{\mathcal{D}}U_{ij} \leq (n - 1)d$.*

Proof. By Corollary 3.2, we know that the row reduced form of V is I_n . Moreover, since $I_n = UV$, we can compute the degree bound of U by using Fact 3.1. Clearly,

$$\vec{v} + (|\vec{u}| - |\vec{v}| - \min_j\{u_j\})\vec{e} \leq \vec{v} + (|\vec{u}| - \min_j\{u_j\})\vec{e},$$

where $\vec{u} := \max(\vec{0}, \text{rowdeg}V)$ and $\vec{v} := \max(\vec{0}, \text{rowdeg}I_n) = \vec{0}$. Since the degree of each row of V is bounded by d , $(|\vec{u}| - \min_j\{u_j\}) \leq (n - 1)d$. Then, by Fact 3.1, $\text{rowdeg}U \leq (n - 1)d$. Therefore, $\deg_{\mathcal{D}}U_{ij} \leq (n - 1)d$. □

Corollary 3.4. *Let H be same as that in Theorem 3.3. Then $\deg_{\mathcal{D}}H_{ij} \leq nd$.*

Proof. Since $\deg_{\mathcal{D}}U_{ij} \leq (n - 1)d$ and $\deg_{\mathcal{D}}A_{ij} \leq d$, $\deg_{\mathcal{D}}H_{ij} \leq nd$. □

4 Computing Hermite Forms by Linear Systems over $F(t)$

In this section we present our polynomial-time algorithm to compute the Hermite form of a matrix over $F(t)[\mathcal{D}; \delta]$. We exhibit a variant of the linear system method developed in Kaltofen et al. (1987) and Storjohann (1994). The approach of these papers is to reduce the problem of computing the Hermite of matrices with (usual) polynomial entries in $F[z]$ to the problem of solving a linear system equations over F . Analogously, we reduce the problem of computing the Hermite form over $F[t][\mathcal{D}; \delta]$ to solving linear systems over $F(t)$. The point is that the field $F(t)$ over which we solve is the usual, commutative, field of rational functions.

For convenience, we assume that our matrix is over $F[t][\mathcal{D}; \delta]$ instead of $F(t)[\mathcal{D}; \delta]$, which can easily be achieved by clearing denominators with a “scalar” multiple from $F[t]$. This is clearly a unimodular operation in the class of matrices over $F(t)[\mathcal{D}; \delta]$.

We first consider formulating the computation of the Hermite form a matrix over $F(t)[\mathcal{D}; \delta]$ as the solution of a “pseudo”-linear system over $F(t)[\mathcal{D}; \delta]$ (i.e., a matrix equation over the non-commutative ring $F(t)[\mathcal{D}; \delta]$).

Theorem 4.1. *Let $A \in F[t][\mathcal{D}; \delta]^{n \times n}$ have full row rank, with $\deg_{\mathcal{D}}A_{i,j} \leq d$, and $(d_1, \dots, d_n) \in \mathbb{N}^n$ be given. Consider the system of equations $PA = G$, for $n \times n$ matrices for $P, G \in F(t)[\mathcal{D}; \delta]$ restricted as follows:*

- The degree (in \mathcal{D}) of each entry of P is bounded by $(n - 1)d + \max_{1 \leq i \leq n} d_i$.
- The matrix G is upper triangular, where every diagonal entry is monic and the degree of each off-diagonal entry is less than the degree of the diagonal entry below it.
- The degree of the i th diagonal entry of G is d_i .

Let H be the Hermite form of A and $(h_1, \dots, h_n) \in \mathbb{N}^n$ be the degrees of the diagonal entries of H . Then the following are true:

- (a) There exists at least one pair P, G as above with $PA = G$ if and only if $d_i \geq h_i$ for $1 \leq i \leq n$.
- (b) If $d_i = h_i$ for $1 \leq i \leq n$ then G is the Hermite form of A and P is a unimodular matrix.

Proof. The proof is similar to that of [Kaltofen et al. \(1987\)](#), Lemma 2.1. Given a degree vector (d_1, \dots, d_n) , we view $PA = G$ as a system of equations in the unknown entries of P and G . Since H is the Hermite form of A , there exist a unimodular matrix U such that $UA = H$. Thus $PU^{-1}H = G$ and the matrix PU^{-1} must be upper triangular since the matrices H and G are upper triangular. Moreover, since the matrix PU^{-1} is in $\mathbb{F}(t)[\mathcal{D}; \delta]^{n \times n}$, and $G_{ii} = (PU^{-1})_{ii} \cdot H_{ii}$ for $1 \leq i \leq n$, we know $d_i \geq h_i$ for $1 \leq i \leq n$. For the other direction, we suppose $d_i \geq h_i$ for $1 \leq i \leq n$. Let $D = \text{diag}(\mathcal{D}^{d_1-h_1}, \dots, \mathcal{D}^{d_n-h_n})$. Then since $(DU)A = (DH)$, we can set $P = DU$ and $G = DH$ as a solution to $PA = G$, and the i th diagonal of G has degree d_i by construction. By Corollary [3.3](#), we know $\text{deg}_{\mathcal{D}} U_{i,j} \leq (n - 1)d$ and so $\text{deg}_{\mathcal{D}} P_{i,j} \leq (n - 1)d + \max_{1 \leq i \leq n} d_i$.

To prove (b), suppose $d_i = h_i$ for $1 \leq i \leq n$ and that, contrarily, G is *not* the Hermite form of A . Since PU^{-1} is an upper triangular matrix with ones on the diagonal, PU^{-1} is a unimodular matrix. Thus P is a unimodular matrix and, by Corollary [3.1](#), G is the (unique) Hermite form of A , a contradiction. \square

Lemma 4.1. *Let $A, P, (d_1, \dots, d_n)$, and G be as in Theorem [4.1](#), and let $\beta := (n - 1)d + \max_{1 \leq i \leq n} d_i$. Also, assume that $\text{deg}_t A_{ij} \leq e$ for $1 \leq i, j \leq n$. Then we can express the system $PA = G$ as a linear system over $\mathbb{F}(t)$ as $\widehat{P}\widehat{A} = \widehat{G}$ where*

$$\widehat{P} \in \mathbb{F}(t)^{n \times n(\beta+1)}, \quad \widehat{A} \in \mathbb{F}[t]^{n(\beta+1) \times n(\beta+d+1)}, \quad \widehat{G} \in \mathbb{F}(t)^{n \times n(\beta+d+1)}.$$

Assuming the entries \widehat{A} are known while the entries of \widehat{P} and \widehat{G} are indeterminates, the system of equations from $\widehat{P}\widehat{A} = \widehat{G}$ for the entries of \widehat{P} and \widehat{G} is linear over $\mathbb{F}(t)$ in its unknowns, and the number of equations and unknowns is $O(n^3d)$. The entries in \widehat{A} are in $\mathbb{F}[t]$ and have degree at most e .

Proof. Since $\text{deg}_{\mathcal{D}} P_{i,j} \leq \beta$, each entry of P has at most $(\beta + 1)$ coefficients in $\mathbb{F}(t)$ and can be written as $P_{ij} = \sum_{0 \leq k \leq \beta} P_{ijk} \mathcal{D}^k$. We let $\widehat{P} \in \mathbb{F}(t)^{n \times n(\beta+1)}$ be the matrix formed from P with P_{ij} replaced by the row vector $(P_{ij0}, \dots, P_{ij\beta}) \in \mathbb{F}(t)$.

Since $\text{deg}_{\mathcal{D}} P \leq \beta$, when forming PA , the entries in A are multiplied by \mathcal{D}^ℓ for $0 \leq \ell \leq \beta$, resulting in polynomials of degree in \mathcal{D} of degree at most $\mu = \beta + d$.

Thus, we construct \widehat{A} as the matrix formed from A with A_{ij} replaced by the $(\beta + 1) \times (\mu + 1)$ matrix whose ℓ th row is

$$(A_{ij0}^{[\ell]}, A_{ij1}^{[\ell]}, \dots, A_{ij\mu}^{[\ell]}) \text{ such that } \mathcal{D}^\ell A_{ij} = A_{ij0}^{[\ell]} + A_{ij1}^{[\ell]} \mathcal{D} + \dots + A_{ij\mu}^{[\ell]} \mathcal{D}^\mu.$$

Note that by Lemma 2.1 we can compute $\mathcal{D}^\ell A_{i,j}$ quickly.

Finally, we construct the matrix \widehat{G} . Each entry of G has degree in \mathcal{D} of degree at most $nd \leq n(\beta + d + 1)$. Thus, initially \widehat{G} is the matrix formed by G with G_{ij} replaced by

$$(G_{ij0}, \dots, G_{ij\mu}) \text{ where } G_{ij} = G_{ij0} + G_{ij1} \mathcal{D} + \dots + G_{ij\mu} \mathcal{D}^\mu.$$

However, because of the structure of the system we can fix values of many of the entries of \widehat{G} as follows. First, since every diagonal entry of the Hermite form is monic, we know the corresponding entry in \widehat{G} is 1. Also, by Corollary 3.4 the degree in \mathcal{D} of every diagonal entry of H is bounded by nd , and every off-diagonal has degree in \mathcal{D} less than that of the diagonal below it (and hence less than nd), and we can set all coefficients of larger powers of \mathcal{D} to 0 in \widehat{G} .

The resulting system $\widehat{P}\widehat{A} = \widehat{G}$, restricted as above according to Theorem 4.1, has $O(n^3d)$ linear equations in $O(n^3d)$ unknowns. Since the coefficients in \widehat{A} are all of the form $\mathcal{D}^\ell A_{ij}$, and since this does not affect their degree in t , the degree in t of entries of \widehat{A} is the same as that of A , namely e . □

With more work, we believe the dimension of the system can be reduced to $O(n^2d) \times O(n^2d)$ if we apply the techniques presented in Storjohann (1994) Section 4.3, wherein the unknown coefficients of \widehat{G} are removed from the system. See also Labhalla et al. (1996).

So far, we have shown how to convert the differential system over $F(t)[\mathcal{D}; \delta]$ into a linear system over $F(t)$. Also, we note, by Theorem 4.1, that the correct degree of the i th diagonal entry in the Hermite form of A can be found by seeking the smallest non-negative integer k such that $PA = G$ is consistent when $\deg_{\mathcal{D}} G_{j,j} = nd$ for $j = 1, \dots, i - 1, i + 1, \dots, n$ and $k \leq \deg_{\mathcal{D}} G_{i,i}$. Using binary search, we can find the correct degrees of all diagonal entries by solving at most $O(n \log(nd))$ systems. We then find the correct degrees of the diagonal entries in the Hermite form of A , solving the system $PA = G$ with the correct diagonal degrees gives the matrices U and H such that $UA = H$ where H is the Hermite form of A .

Theorem 4.2. *Let $A \in F[t][\mathcal{D}; \delta]^{n \times n}$ with $\deg_{\mathcal{D}} A_{ij} \leq d$ and $\deg_t A_{ij} \leq e$ for $1 \leq i, j \leq n$. Then we can compute the Hermite form $H \in F(t)[\mathcal{D}; \delta]$ of A , and a unimodular $U \in F[t][\mathcal{D}; \delta]$ such that $UA = H$, with $O((n^{10}d^3 + n^7d^2e) \log(nd))$ operations in F*

Proof. Lemma 4.1 and the following discussion, above shows that computing U and H is reduced to solving $O(n \log(nd))$ systems of linear equations over $F(t)$, each of which is $m \times m$ for $m = O(n^3d)$ and in which the entries have degree e . Using standard linear algebra this can be solved with $O(m^4e)$ operations in F , since any solution has degree at most me (see von zur Gathen and Gerhard

(2003)). A somewhat better strategy is to use the t -adic lifting approach of Dixon (1982), which would require $O(m^3 + m^2e)$ operations in F for each system, giving a total cost of $O((n^{10}d^3 + n^7d^2e) \log(nd))$ operations in F . \square

As noted above, it is expected that we can bring this cost down through a smaller system similar to that of Storjohann (1994), to a cost of $O((n^7d^2 + n^5d^2e) \log(nd))$. Nonetheless, the algorithm as it is stated achieves a guaranteed polynomial-time solution.

It is often the case that we are considering differential systems over $\mathbb{Q}(t)[\mathcal{D}; \delta]$, where we must contend with growth in coefficients in \mathcal{D} , t and in the size of the rational coefficients. However, once again we may employ the fact that the Hermite form and unimodular transformation matrix are solutions of a linear system over $\mathbb{Q}[t]$. For convenience, we can assume in fact that our input is in $\mathbb{Z}[t][\mathcal{D}; \delta]^{n \times n}$ (since the rational matrix to eliminate denominators is unimodular in $\mathbb{Q}(t)[\mathcal{D}; \delta]$). There is some amount of extra coefficient growth when going from A to \hat{A} ; namely we take up to nd derivatives, introducing a multiplicative constant of size around $\min((nd)!, e!)$. In terms of the bit-length of the coefficients, this incurs a multiplicative blow-up of only $O(\ell \log(\ell))$ where $\ell = \min(nd, e)$. It follows that we can find the Hermite form of $A \in \mathbb{Q}(t)[\mathcal{D}; \delta]^{n \times n}$ in time polynomial in n , $\deg_t A_{ij}$, $\deg_{\mathcal{D}} A_{ij}$, and $\log \|A_{ij}\|$, the maximum coefficient length in an entry, for $1 \leq i, j \leq n$. A modular algorithm, for example along the lines of Li and Nemes (1997), would improve performance considerably, as might p -adic solvers and a more careful construction of the linear system.

5 Conclusions and Future Work

We have shown that the problem of computing the Hermite form of a matrix over $F(t)[\mathcal{D}; \delta]$ can be accomplished in polynomial time. Moreover, our algorithm will also control growth in coefficient bit-length when $F = \mathbb{Q}$. We have also shown that the degree bounds on Hermite forms in the differential ring are very similar to the regular polynomial case. From a practical point of view our method is still expensive. Our next work will be to investigate more efficient algorithms. We have suggested ways to compress the system of equations and to employ structured matrix techniques. Also, the use of randomization has been shown to be highly beneficial over $F[t]$, and should be investigated in this domain. Finally, our approach should be applicable to difference polynomials and more general Ore polynomial rings.

References

- Abramov, S., Bronstein, M.: On solutions of linear functional systems. In: Proc. ACM International Symposium on Symbolic and Algebraic Computation, pp. 1–7 (2001)
- Beckermann, B., Cheng, H., Labahn, G.: Fraction-free row reduction of matrices of ore polynomials. *Journal of Symbolic Computation* 41(1), 513–543 (2006)

- Bostan, A., Chyzak, F., Le Roux, N.: Products of ordinary differential operators by evaluation and interpolation. In: Proc. International Symposium on Symbolic and Algebraic Computation, pp. 23–30 (2008)
- Bronstein, M., Petkovšek, M.: On Ore rings, linear operators and factorisation. *Programirovanie* 20, 27–45 (1994)
- Cheng, H.: Algorithms for Normal Forms for Matrices of Polynomials and Ore Polynomials. PhD thesis, University of Waterloo (2003), <http://www.cs.uleth.ca/~cheng/publications.html>
- Davies, P., Cheng, H., Labahn, G.: Computing Popov form of general Ore polynomial matrices. In: Milestones in Computer Algebra, pp. 149–156 (2008)
- Dickson, L.E.: Algebras and their arithmetics. G.E. Stechert, New York (1923)
- Jean Dieudonné, M.: Les déterminants sur un corps non commutatif. *Bulletin de la Société Mathématique de France* 71, 27–45 (1943)
- Dixon, J.D.: Exact solution of linear equations using p -adic expansions. *Numer. Math.* 40, 137–141 (1982)
- von zur Gathen, J., Gerhard, J.: *Modern Computer Algebra*. Cambridge University Press, Cambridge (2003)
- Hermite, C.: Sur les fonctions de sept lettres. *C.R. Acad. Sci. Paris* 57, 750–757 (1863); *OEuvres*. Gauthier-Villars, Paris 2, 280–288 (1908)
- van der Hoeven, J.: FFT-like multiplication of linear differential operators. *Journal of Symbolic Computation* 33(1), 123–127 (2002)
- Jacobson, N.: *The Theory of Rings*. American Math. Soc., New York (1943)
- Kaltofen, E., Krishnamoorthy, M.S., Saunders, B.D.: Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM J. Algebraic and Discrete Methods* 8, 683–690 (1987)
- Kannan, R.: Polynomial-time algorithms for solving systems of linear equations over polynomials. *Theoretical Computer Science* 39, 69–88 (1985)
- Labhalla, S., Lombardi, H., Marlin, R.: Algorithmes de calcul de la réduction de Hermite d’une matrice à coefficients polynomiaux. *Theoretical Computer Science* 161(1–2), 69–92 (1996)
- Li, Z.: A subresultant theory for Ore polynomials with applications. In: Proc. International Symposium on Symbolic and Algebraic Computation, pp. 132–139 (1998)
- Li, Z., Nemes, I.: A modular algorithm for computing greatest common right divisors of ore polynomials. In: *ISSAC 1997: Proceedings of the 1997 international symposium on symbolic and algebraic computation*. ACM, New York (1997)
- Middeke, J.: A polynomial-time algorithm for the jacobson form for matrices of differential operators. Technical Report 08-13, Research Institute for Symbolic Computation (RISC), Linz, Austria (2008)
- Mulders, T., Storjohann, A.: On lattice reduction for polynomial matrices. *Journal of Symbolic Computation* 35(4), 377–401 (2003)
- Popov, V.: Invariant description of linear, time-invariant controllable systems. *SIAM J. Control* 10, 252–264 (1972)
- Smith, H.J.S.: On systems of linear indeterminate equations and congruences. *Philos. Trans. Royal Soc. London* 151, 293–326 (1861)
- Storjohann, A.: Computation of Hermite and Smith normal forms of matrices. Master’s thesis, University of Waterloo (1994)
- Villard, G.: Generalized subresultants for computing the smith normal form of polynomial matrices. *Journal of Symbolic Computation* 20, 269–286 (1995)
- Wedderburn, J.H.M.: Non-commutative domains of integrity. *Journal für die reine und angewandte Mathematik* 167, 129–141 (1932)

On the Computation of Comprehensive Boolean Gröbner Bases

Shutaro Inoue

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

Abstract. We show that a comprehensive Boolean Gröbner basis of an ideal I in a Boolean polynomial ring $\mathbf{B}(\bar{A}, \bar{X})$ with main variables \bar{X} and parameters \bar{A} can be obtained by simply computing a usual Boolean Gröbner basis of I regarding both \bar{X} and \bar{A} as variables with a certain block term order such that $\bar{X} \gg \bar{A}$. The result together with a fact that a finite Boolean ring is isomorphic to a direct product of the Galois field \mathbb{GF}_2 enables us to compute a comprehensive Boolean Gröbner basis by only computing corresponding Gröbner bases in a polynomial ring over \mathbb{GF}_2 . Our implementation in a computer algebra system Risa/Asir shows that our method is extremely efficient comparing with existing computation algorithms of comprehensive Boolean Gröbner bases.

1 Introduction

A commutative ring \mathbf{B} with an identity is called a *Boolean ring* if every element of which is idempotent. A residue class ring $\mathbf{B}[X_1, \dots, X_n]/\langle X_1^2 - X_1, \dots, X_n^2 - X_n \rangle$ with an ideal $\langle X_1^2 - X_1, \dots, X_n^2 - X_n \rangle$ also becomes a Boolean ring, which is called a *Boolean polynomial ring* and denoted by $\mathbf{B}(X_1, \dots, X_n)$. A Gröbner basis in a Boolean polynomial ring (called a *Boolean Gröbner basis*) is first introduced in [5,6] and further developments are done in [7,8,10,12]. The original computation algorithm introduced in [5,6] uses a special monomial reduction which is more complicated than a usual monomial reduction in a polynomial ring over a field. Though the algorithm is based on such a complicated monomial reduction, it is also directly applicable for the computations of comprehensive Boolean Gröbner bases. This algorithm is implemented in [7] for computations of both Boolean Gröbner bases and comprehensive Boolean Gröbner bases. In [9], an alternative algorithm is introduced where we can obtain a Boolean Gröbner basis by only computing usual Gröbner bases in a polynomial ring over the Galois field \mathbb{GF}_2 . Its implementation brought us a much faster program than the previous one of [7]. Unfortunately, however, this algorithm is not applicable for the computations of comprehensive Boolean Gröbner bases.

In this paper, we show that a Boolean Gröbner basis $\{g_1(\bar{A}, \bar{X}), \dots, g_s(\bar{A}, \bar{X})\}$ of an ideal I in a Boolean polynomial ring $\mathbf{B}(\bar{A}, \bar{X})$ with variables $\bar{A} = A_1, \dots, A_m$ and $\bar{X} = X_1, \dots, X_n$ w.r.t. a block term order such that $\bar{X} \gg \bar{A}$ is stable for any specialization of \bar{A} , i.e. $\{g_1(\bar{a}, \bar{X}), \dots, g_s(\bar{a}, \bar{X})\}$ also becomes a Boolean Gröbner

basis of the ideal $\langle \{f(\bar{a}, \bar{X}) \mid f(\bar{A}, \bar{X}) \in I\} \rangle$ in a Boolean polynomial ring $\mathbf{B}'(\bar{X})$ for any Boolean extension \mathbf{B}' of \mathbf{B} and elements $\bar{a} = a_1, \dots, a_m$ of \mathbf{B}' . This result together with an algorithm of [9] enables us to compute comprehensive Boolean Gröbner bases by only computing usual Gröbner bases in polynomial rings over the Galois field \mathbb{GF}_2 . We implemented the method in a computer algebra system Risa/Asir ([4]). Our program achieves a tremendous speedup comparing with the previous one. It enables us to do our recent work [11] of a non-trivial application of Boolean Gröbner bases.

The paper is organized as follows. In section 2, we give a quick review of Boolean Gröbner bases, which we need for understanding our work. More detailed descriptions can be found in [12]. We also prove a key fact (namely Theorem 16) which plays an important role in this paper. In section 3, we describe the algorithm to compute Boolean Gröbner bases introduced in [9]. In section 4, we prove the above result in much stronger form. In section 5, we describe our implementation together with some timing data of our computation experiments.

2 Boolean Gröbner Bases

A Boolean Gröbner basis is defined as a natural modification of a Gröbner basis in a Boolean polynomial ring. Though it was introduced in [5,6] together with a computation algorithm using a special monomial reduction, the same notion was independently discovered in [15] for a polynomial ring over a more general coefficient ring. In this section, we give a quick review of Boolean Gröbner bases. More detailed descriptions with proofs can be found in [8] or [15].

2.1 Boolean Polynomial Ring

Definition 1. *A commutative ring \mathbf{B} with an identity 1 is called a Boolean ring if every element a of \mathbf{B} is idempotent, i.e. $a^2 = a$.*

$\langle \mathbf{B}, \vee, \wedge, \neg \rangle$ becomes a Boolean algebra with the Boolean operations \vee, \wedge, \neg defined by $a \vee b = a + b + a \cdot b, a \wedge b = a \cdot b, \neg a = 1 + a$. Conversely, for a Boolean algebra $\langle \mathbf{B}, \vee, \wedge, \neg \rangle$, if we define $+$ and \cdot by $a + b = (\neg a \wedge b) \vee (a \wedge \neg b)$ and $a \cdot b = a \wedge b$, $\langle \mathbf{B}, +, \cdot \rangle$ becomes a Boolean ring.

Since $\neg a = a$ in a Boolean ring, we do not need to use the symbol ' \neg ', however, we also use $-$ when we want to stress its meaning.

Definition 2. *Let \mathbf{B} be a Boolean ring. A quotient ring $\mathbf{B}[X_1, \dots, X_n] / \langle X_1^2 - X_1, \dots, X_n^2 - X_n \rangle$ modulo an ideal $\langle X_1^2 - X_1, \dots, X_n^2 - X_n \rangle$ becomes a Boolean ring. It is called a Boolean polynomial ring and denoted by $\mathbf{B}(X_1, \dots, X_n)$, its element is called a Boolean polynomial.*

Note that a Boolean polynomial of $\mathbf{B}(X_1, \dots, X_n)$ is uniquely represented by a polynomial of $\mathbf{B}[X_1, \dots, X_n]$ that has at most degree 1 for each variable X_i . In what follows, we identify a Boolean polynomial with such a representation.

Multiple variables such as A_1, \dots, A_m or X_1, \dots, X_n are abbreviated to \bar{A} or \bar{X} respectively. Lower small roman letters such as a, b, c are usually used for

elements of a Boolean ring \mathbf{B} . The symbol \bar{a} denotes an m -tuple of element of \mathbf{B} for some m . For a Boolean polynomial $f(\bar{A}, \bar{X})$ with variables \bar{A} and \bar{X} , $f(\bar{a}, \bar{X})$ denotes a Boolean polynomial in $\mathbf{B}[\bar{X}]$ obtained by specializing \bar{A} with \bar{a} .

2.2 Gröbner Bases

In what follows, we assume that some term order on a set of power products of variables is given. For a polynomial f in a polynomial ring $\mathbf{B}[\bar{X}]$ over a Boolean ring \mathbf{B} , we use the notations $LT(f)$, $LM(f)$ and $LC(f)$ to denote the leading power product, the leading monomial and leading coefficient of f respectively. $f - LM(f)$ is also denoted by $Rd(f)$. We also use the notations $LT(F)$ and $LM(F)$ to denote the sets $\{LT(f)|f \in F\}$ and $\{LM(f)|f \in F\}$ for a (possibly infinite) subset F of $\mathbf{B}[\bar{X}]$. $T(\bar{X})$ denotes the set of power products consisting of variables \bar{X} .

Definition 3. For an ideal I of a polynomial ring $\mathbf{B}[\bar{X}]$, a finite subset G of I is called a Gröbner basis of I if $\langle LM(I) \rangle = \langle LM(G) \rangle$.

Definition 4. For a polynomial $f \in \mathbf{B}[\bar{X}]$, let $a = LC(f)$, $t = LT(f)$ and $h = Rd(f)$. A monomial reduction \rightarrow_f by f is defined as follows:

$$bts + p \rightarrow_f (1 - a)bs + absh + p.$$

(Note that $(bts + p) - ((1 - a)bs + absh + p) = bs(af)$.)

Where s is a term of $T(\bar{X})$, b is an element of \mathbf{B} such that $ab \neq 0$ and p is any polynomial of $\mathbf{B}[\bar{X}]$. For a set $F \subseteq \mathbf{B}[\bar{X}]$, we write $g \rightarrow_F g'$ if and only if $g \rightarrow_f g'$ for some $f \in F$. A recursive closure of \rightarrow_F is denoted by $\overset{*}{\rightarrow}_F$, i.e. $g \overset{*}{\rightarrow}_F g'$ if and only if $g = g'$ or there exist a sequence of monomial reductions $g \rightarrow_F g_1 \rightarrow_F \dots \rightarrow_F g_k = g'$.

Theorem 5. When F is finite, \rightarrow_F is noetherian, that is there is no infinite sequence of polynomials g_1, g_2, \dots such that $g_i \rightarrow_F g_{i+1}$ for each $i = 1, 2, \dots$

Theorem 6. Let I be an ideal of a polynomial ring $\mathbf{B}[\bar{X}]$.

A finite subset G of I is a Gröbner basis of I if and only if $\forall h \in I \ h \overset{*}{\rightarrow}_G 0$.

Using our monomial reductions, a reduced Gröbner basis is similarly defined as in a polynomial ring over a field. A Gröbner basis G is *reduced* if each polynomial of G is not reducible by a monomial reduction of any other polynomial of G . In a polynomial ring over a field, a reduced Gröbner basis is uniquely determined. In our case, however, this property does not hold.

Example 1. Let $\mathbf{B} = \mathbb{GF}_2 \times \mathbb{GF}_2$. In a polynomial ring $\mathbf{B}[X]$, $\{(1, 0)X, (0, 1)X\}$ and $\{(1, 1)X\}$ are both reduced Gröbner bases of the same ideal.

In order to have a unique Gröbner basis, we need one more definition.

Definition 7. A reduced Gröbner basis G is said to be stratified if G does not contain two polynomials which have the same leading power product.

Theorem 8. If G and G' are stratified Gröbner bases of the same ideal w.r.t. some term order, then $G = G'$.

In the above example, $\{(1, 1)X\}$ is the stratified Gröbner basis, but the other is not.

Definition 9. For a polynomial f , $LC(f)f$ is called the Boolean closure of f , and denoted by $bc(f)$. If $f = bc(f)$, f is said to be Boolean closed.

Theorem 10. Let G be a Gröbner basis of an ideal I , then $bc(G) \setminus \{0\}$ is also a Gröbner basis of an ideal I .

Theorem 11. Let G be a reduced Gröbner basis, then every element is Boolean closed.

S -polynomial is also defined similarly as in a polynomial ring over a field.

Definition 12. Let $f = atr + f'$ and $g = bsr + g'$ be polynomials where $a = LC(f)$, $b = LC(g)$, $tr = LT(f)$ and $sr = LT(g)$ for some power product t, s, r such that $GCD(t, s) = 1$, i.e. t and s do not contain a common variable. The polynomial $bsf + atg = bsf' + atg'$ is called an S -polynomial of f and g and denoted by $S(f, g)$.

As in a polynomial ring over a field, the following property is crucial for the construction of Gröbner bases.

Theorem 13. Let G be a finite set of polynomials such that each element of G is Boolean closed. Then, G is a Gröbner basis if and only if $S(f, g) \xrightarrow{*}_G 0$ for any pair f, g of G .

For any given finite set F , using our monomial reductions, we can always construct a Gröbner basis of $\langle F \rangle$ by computing Boolean closures and S -polynomials using the following algorithms. It is also easy to construct a stratified Gröbner basis from a Gröbner basis.

Algorithm BC

Input: F a finite subset of $\mathbf{B}[\bar{X}]$

Output: F' a set of Boolean closed polynomials such that $\langle F' \rangle = \langle F \rangle$

begin

$F' = \emptyset$

while there exists a polynomial $f \in F$ which is not Boolean closed

$F = F \cup \{bc(f) - f\} \setminus \{f\}$, $F' = F' \cup \{bc(f)\}$

end.

Algorithm GBasis**Input:** F a finite subset of $\mathbf{B}[\bar{X}]$, $>$ a term order of $T(\bar{X})$ **Output:** G a Gröbner basis of $\langle F \rangle$ w.r.t. $>$

begin

 $G = \text{BC}(F)$ while there exists two polynomials $p, q \in G$ such that $S(p, q) \xrightarrow{*}_G h$ for some non-zero polynomial h which is irreducible by \rightarrow_G $G = G \cup \text{BC}(\{h\})$

end.

Since any element of a Boolean ring is idempotent, a Boolean polynomial ring is more natural to work on. We can also define Gröbner bases in Boolean polynomial rings. A power product $X_1^{l_1} \cdots X_n^{l_n}$ is called a *Boolean power product* if each l_i is either 0 or 1. The set of all Boolean power products consisting of variables \bar{X} is denoted by $BT(\bar{X})$. A Boolean polynomial $f(\bar{X})$ in $\mathbf{B}(\bar{X})$ is uniquely represented by $b_1 t_1 + \cdots + b_k t_k$ with elements b_1, \dots, b_k of \mathbf{B} and distinct Boolean power products t_1, \dots, t_k . We call $b_1 t_1 + \cdots + b_k t_k$ the *canonical representation* of $f(\bar{X})$. Since $BT(\bar{X})$ is a subset of $T(\bar{X})$, a term order $>$ on $T(\bar{X})$ is also defined on $BT(\bar{X})$. Given a term order $>$, we use the same notations $LT(f)$, $LM(f)$, $LC(f)$ and $Rd(f)$ as before, which are defined by using its canonical representation. We also use the same notations $LT(F)$ and $LM(F)$ for a set F of Boolean polynomials as before.

Definition 14. For an ideal I of a Boolean polynomial ring $\mathbf{B}(\bar{X})$, a finite subset G of I is called a Boolean Gröbner basis of I if $\langle LM(I) \rangle = \langle LM(G) \rangle$ in $\mathbf{B}(\bar{X})$.

Using canonical representations of Boolean polynomials, we can also define monomial reductions for Boolean polynomials as Definition 4 and have the same property of Theorem 6. We can also define a stratified Boolean Gröbner basis as in Definition 7, which is unique w.r.t. a term order. The Boolean closure of a Boolean polynomial is also similarly defined as Definition 9 and the same properties of Theorem 10, 11 and 13 hold. Construction of a Boolean Gröbner basis is very simple. Given a finite set of Boolean polynomials $F \subseteq \mathbf{B}(\bar{X})$. Compute a Gröbner basis G of the ideal $\langle F \cup \{X_1^2 - X_1, \dots, X_n^2 - X_n\} \rangle$ in $\mathbf{B}[\bar{X}]$ w.r.t. the same term order. Then, $G \setminus \{X_1^2 - X_1, \dots, X_n^2 - X_n\}$ is a Boolean Gröbner basis of $\langle F \rangle$ in $\mathbf{B}(\bar{X})$. If G is stratified, then $G \setminus \{X_1^2 - X_1, \dots, X_n^2 - X_n\}$ is also stratified.

2.3 Comprehensive Gröbner Bases (Previous Algorithm)

In a polynomial ring over a field, construction of a comprehensive Gröbner basis is not so simple in general. In order to get a uniform (with respect to parameters) representation of reduced Gröbner bases, we need to divide a parameter space into several partitions according to the conditions that parameters satisfy. (See [12, 3, 13, 14, 16].) In our Boolean polynomial ring, however, we can always construct a stratified comprehensive Boolean Gröbner basis. We do not even need to divide a parameter space. In this section, we present an important fact which enables us to have a naive construction method of comprehensive

Boolean Gröbner bases. This method is implemented in [719], although a detailed description of the fact is recently published in [12]. We prove it in a more general form, which plays an important role in this paper.

We use variables $\bar{A} = A_1, \dots, A_m$ for parameters and variables $\bar{X} = X_1, \dots, X_n$ for main variables. We also assume that some term order on $T(\bar{X})$ is given.

Definition 15. *Let I be an ideal in $\mathbf{B}(\bar{A})$. For a Boolean extension \mathbf{B}' of \mathbf{B} , i.e. a Boolean ring which includes \mathbf{B} as a subring, $V_{\mathbf{B}'}(I)$ denotes the variety of I in \mathbf{B}' , i.e. $V_{\mathbf{B}'}(I) = \{\bar{a} \in \mathbf{B}'^m \mid \forall f \in I f(\bar{a}) = 0\}$. Let $F = \{f_1(\bar{A}, \bar{X}), \dots, f_l(\bar{A}, \bar{X})\}$ be a finite subset of $\mathbf{B}(\bar{A}, \bar{X})$. A finite subset $G = \{g_1(\bar{A}, \bar{X}), \dots, g_k(\bar{A}, \bar{X})\}$ of $\mathbf{B}(\bar{A}, \bar{X})$ is called a comprehensive Boolean Gröbner basis of F on I , if $G(\bar{a}) = \{g_1(\bar{a}, \bar{X}), \dots, g_k(\bar{a}, \bar{X})\} \setminus \{0\}$ is a Boolean Gröbner basis of the ideal $\langle F(\bar{a}) \rangle = \langle f_1(\bar{a}, \bar{X}), \dots, f_l(\bar{a}, \bar{X}) \rangle$ in $\mathbf{B}'(\bar{X})$ for any Boolean extension \mathbf{B}' of \mathbf{B} and $\bar{a} \in V_{\mathbf{B}'}(I)$. When $I = \{0\}$, we simply call G a comprehensive Boolean Gröbner basis of F . G is also said to be stratified if $G(\bar{a})$ is stratified for any $\bar{a} \in V_{\mathbf{B}'}(I)$.*

Theorem 16. *Let I be an ideal in $\mathbf{B}(\bar{A})$. Let F be a finite subset of $\mathbf{B}(\bar{A}, \bar{X})$. Considering each $f_i(\bar{A}, \bar{X})$ as a member of the Boolean polynomial ring $(\mathbf{B}(\bar{A})/I)(\bar{X})$ over the coefficient Boolean ring $\mathbf{B}(\bar{A})/I$, let G be a (stratified) Boolean Gröbner basis of the ideal $\langle F \rangle$ in this polynomial ring, then G is a (stratified) comprehensive Boolean Gröbner basis of F on I .*

proof. Note first that G is also a (stratified) Boolean Gröbner basis of $\langle F \rangle$ in $(\mathbf{B}'(\bar{A})/I)(\bar{X})$, here we abuse the same symbol I to denote an ideal of $\mathbf{B}'(\bar{A})$ generated from I . Therefore, it suffices to consider only specialization from $V_{\mathbf{B}}(I)$. Let \bar{a} be an arbitrary m -tuple lying on $V_{\mathbf{B}}(I)$. Note that the specialization of parameters \bar{A} with \bar{a} induces a homomorphism from $(\mathbf{B}(\bar{A})/I)(\bar{X})$ to $\mathbf{B}(\bar{X})$. We clearly have $\langle F(\bar{a}) \rangle = \langle G(\bar{a}) \rangle$ in $\mathbf{B}(\bar{X})$.

If $f(\bar{A}, \bar{X}) \rightarrow_{g(\bar{A}, \bar{X})} h(\bar{A}, \bar{X})$ in $(\mathbf{B}(\bar{A})/I)(\bar{X})$, then $f(\bar{A}, \bar{X}) = p(\bar{A})ts + f'(\bar{A}, \bar{X})$, $g(\bar{A}, \bar{X}) = q(\bar{A})t + g'(\bar{A}, \bar{X})$ and $h(\bar{A}, \bar{X}) = (1 - q(\bar{A}))p(\bar{A})ts + q(\bar{A})p(\bar{A})sg'(\bar{A}, \bar{X}) + f'(\bar{A}, \bar{X})$ for some $t, s \in T(\bar{X})$ and $p(\bar{A}), q(\bar{A}) \in \mathbf{B}(\bar{A})/I$ and $f'(\bar{A}, \bar{X}), g'(\bar{A}, \bar{X}) \in (\mathbf{B}(\bar{A})/I)(\bar{X})$, where $q(\bar{A})t$ is the leading monomial of $g(\bar{A}, \bar{X})$.

In case $q(\bar{a})p(\bar{a}) \neq 0$, certainly $q(\bar{a}) \neq 0$ and $p(\bar{a}) \neq 0$, so $q(\bar{a})t$ is the leading monomial of $g(\bar{a}, \bar{X})$ and $p(\bar{a})ts$ is a monomial of $f(\bar{A}, \bar{X})$ and $f(\bar{a}, \bar{X}) \rightarrow_{g(\bar{a}, \bar{X})} h(\bar{a}, \bar{X})$. Otherwise, $h(\bar{a}, \bar{X}) = f(\bar{a}, \bar{X})$. In either case, we have $f(\bar{a}, \bar{X}) \xrightarrow{*}_{g(\bar{a}, \bar{X})} h(\bar{a}, \bar{X})$. Therefore, if $f(\bar{A}, \bar{X}) \rightarrow_G h(\bar{A}, \bar{X})$ in $(\mathbf{B}(\bar{A})/I)(\bar{X})$, then we have $f(\bar{a}, \bar{X}) \xrightarrow{*}_{G(\bar{a})} h(\bar{a}, \bar{X})$ in $\mathbf{B}(\bar{X})$. Any Boolean polynomial in the ideal $\langle F(\bar{a}) \rangle$ is equal to $f(\bar{a}, \bar{X})$ for some Boolean polynomial $f(\bar{A}, \bar{X})$ in the ideal $\langle F \rangle$ of $(\mathbf{B}(\bar{A})/I)(\bar{X})$. Since G is a Boolean Gröbner basis of $\langle F \rangle$, we have $f(\bar{A}, \bar{X}) \xrightarrow{*}_G 0$. By the above observation, we have $f(\bar{a}, \bar{X}) \xrightarrow{*}_{G(\bar{a})} 0$. This shows that G is a comprehensive Boolean Gröbner basis of F on I .

Suppose G is stratified, then any element g of G is Boolean closed.

So, if $LC(g)(\bar{a}) = 0$, then $g(\bar{a}, \bar{X})$ must be equal to 0. Therefore, unless $g(\bar{a}, \bar{X}) = 0$, we have $LT(g(\bar{a}, \bar{X})) = LT(g(\bar{A}, \bar{X}))$. Now it is clear that $G(\bar{a})$ is stratified. \square

3 Alternative Algorithm

An alternative computation algorithm of Boolean Gröbner bases introduced in [9] is based on the following fact which is essentially a special instance of Theorem 2.3 of [15].

Definition 17. Let \mathbf{B} be a Boolean ring and k be a natural number. \mathbf{B}^k denotes a direct product, i.e. the set of all k -tuples of elements of \mathbf{B} . For an element p of \mathbf{B}^k , $p_i \in \mathbf{B}$ denotes the i -th element of p for each $i = 1, \dots, k$. If we define $p + q$ and $p \cdot q$ for $p, q \in \mathbf{B}^k$ by $(p + q)_i = p_i + q_i$ and $(p \cdot q)_i = p_i \cdot q_i$ for each $i = 1, \dots, k$, \mathbf{B}^k also becomes a Boolean ring. For a polynomial $f(\bar{X})$ in $\mathbf{B}^k[\bar{X}]$ $f_i(i = 1, \dots, k)$ denotes the polynomial in $\mathbf{B}[\bar{X}]$ obtained by replacing each coefficient p of f by p_i . For a Boolean polynomial $f(\bar{X})$ in $\mathbf{B}^k(\bar{X})$, a Boolean polynomial f_i in $\mathbf{B}(\bar{X})$ is defined similarly.

Theorem 18. In a polynomial ring $\mathbf{B}^k[\bar{X}]$, let G be a finite set of Boolean closed polynomials. Then, G is a (reduced) Gröbner basis of an ideal I if and only if $G_i = \{g_i | g \in G\} \setminus \{0\}$ is a (reduced) Gröbner basis of the ideal $I_i = \{f_i | f \in I\}$ in $\mathbf{B}[\bar{X}]$ for each $i = 1, \dots, k$.

Corollary 19. In a Boolean polynomial ring $\mathbf{B}^k(\bar{X})$, let G be a finite set of Boolean closed Boolean polynomials. Then, G is a (reduced) Boolean Gröbner basis of an ideal I if and only if $G_i = \{g_i | g \in G\} \setminus \{0\}$ is a (reduced) Gröbner basis of the ideal $I_i = \{f_i | f \in I\}$ in $\mathbf{B}(\bar{X})$ for each $i = 1, \dots, k$.

Let F be a finite set of polynomials in $\mathbf{B}[\bar{X}]$ for a computable Boolean ring \mathbf{B} . Note that a Boolean subring which is generated from the set of all coefficients of polynomials in F is finite. We denote this ring by \mathbf{B}_F . By Stone's representation theorem, \mathbf{B}_F is isomorphic to \mathbb{GF}_2^k for some natural number k . Let ψ be such an isomorphism. We also extend ψ to a polynomial ring $\mathbf{B}_F[\bar{X}]$. Identifying a polynomial p of $\mathbf{B}[\bar{X}]$ with its image $\psi(p)$ in \mathbb{GF}_2^k , $F_i(i = 1, \dots, k)$ denotes a set of i -th projection of F in $\mathbb{GF}_2[\bar{X}]$, that is $F_i = \{\psi(p)_i | p \in F\}$. For each $i = 1, \dots, k$, let e_i denote an element of \mathbb{GF}_2^k such that its i -th component is 1 and the other component is 0. For a polynomial f in $\mathbb{GF}_2[\bar{X}]$ and each $i = 1, \dots, k$, $Ex_i(f)$ denotes a polynomial in $\mathbb{GF}_2[\bar{X}]^k$ obtained from f by replacing each monomial t with $e_i t$. Now, the algorithm is given as follows:

Algorithm AGBasis

Input: F a finite subset of $\mathbf{B}[\bar{X}]$, $>$ a term order of $T(\bar{X})$

Output: G a Gröbner basis of $\langle F \rangle$ w.r.t. $>$

begin

for each $i = 1, \dots, k$

$G^i = \text{GBasis}(F_i, >)$

$G_i = \{Ex_i(g) | g \in G^i\}$

$G = \psi^{-1}(\cup_{i=1}^k G_i)$

end.

For a finite set F of Boolean polynomials, its Boolean Gröbner basis is also computed by the same algorithm with a suitable modification.

Note that $\mathbf{GBasis}(F_i, >)$ compute a usual Gröbner basis of $\langle F_i \rangle$ in a polynomial ring $\mathbb{GF}_2[\bar{X}]$ over the field \mathbb{GF}_2 . If each G^i is a reduced Gröbner basis, obviously $\cup_{i=1}^k G_i$ is a reduced Gröbner basis and so is G . It is also easy to construct the stratified Gröbner basis from a reduced Gröbner basis. If we can see k is small (say less than 100) and ψ is easily computable from the input F a priori, this algorithm is more practical than the original one as is reported in [9]. When k is very big, however, this algorithm may be impractical. For the construction of a comprehensive Boolean Gröbner basis by the method described in the previous section, we have to compute at least 2^m -many Gröbner bases in $\mathbb{GF}_2[\bar{X}]$ since $\mathbf{B}(\bar{A})$ is isomorphic to \mathbf{B}^{2^m} , which of course is infeasible unless m is small.

Example 2. *The following left constraint with unknown set variables X and Y is equivalent to the right system of equations of a Boolean polynomial ring $\mathbf{B}(X, Y)$, where \mathbf{B} is a Boolean ring that consists of all computable subsets of S (a set of all strings).*

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

Let $F = \{(1 + \{s_1, s_2\})(XY + X + Y), \{s_1\}X + \{s_1\}, \{s_2\}Y + \{s_2\}, XY\}$. \mathbf{B}_F is a finite subring of \mathbf{B} that consists of $\{0, 1, \{s_1\}, \{s_2\}, \{s_1, s_2\}, 1 + \{s_1\}, 1 + \{s_2\}, 1 + \{s_1, s_2\}\}$. It is isomorphic to \mathbb{GF}_3 with an isomorphism ψ given by $\psi(\{s_1\}) = (1, 0, 0)$, $\psi(\{s_2\}) = (0, 1, 0)$ and $\psi(1 + \{s_1, s_2\}) = (0, 0, 1)$. With this isomorphism $F_1 = \{0, X + 1, 0, XY\}$, $F_2 = \{0, 0, Y + 1, XY\}$ and $F_3 = \{XY + X + Y, 0, 0, XY\}$. Reduced Gröbner bases of them in a boolean polynomial ring $\mathbb{GF}_2(X, Y)$ w.r.t. a lexicographic term order such that $X > Y$ are $G^1 = \{Y, X + 1\}$, $G^2 = \{Y + 1, X\}$ and $G^3 = \{X, Y\}$ respectively. So, $G_1 = \{(1, 0, 0)Y, (1, 0, 0)(X + 1)\}$, $G_2 = \{(0, 1, 0)(Y + 1), (0, 1, 0)X\}$ and $G_3 = \{(0, 0, 1)X, (0, 0, 1)Y\}$ and we have a reduced Boolean Gröbner basis $G = \{\{s_1\}Y, \{s_1\}(X + 1), \{s_2\}(Y + 1), \{s_2\}X, (1 + \{s_1, s_2\})X, (1 + \{s_1, s_2\})Y\}$ of F . Stratified Boolean Gröbner basis is obtained simply adding elements which have the same leading monomial. For this G , its stratified Boolean Gröbner basis is $\{X + \{s_1\}, Y + \{s_2\}\}$.

4 Main Result

Let F be a finite set of $\mathbf{B}(\bar{A}, \bar{X})$. As is described in section 2.3, a (stratified) Boolean Gröbner basis G computed in the Boolean polynomial ring $(\mathbf{B}(\bar{A}))(\bar{X})$ becomes a (stratified) comprehensive Boolean Gröbner basis of F . When the \bar{X} -eliminate portion $\langle F \rangle \cap \mathbf{B}(\bar{A})$ (call this I) is not a trivial ideal $\{0\}$, the size of G tends to be extremely big. In most applications, we do not need a specialization by \bar{a} which does not vanish on I . What we actually need is a (stratified) comprehensive Boolean Gröbner basis of F on I . There are essentially two methods to compute I . One is computing a stratified Boolean Gröbner basis of $\langle F \rangle$ in $(\mathbf{B}(\bar{A}))(\bar{X})$, although the computation tends to consume much memory space

as described above. The other is computing a stratified Boolean Gröbner basis of $\langle F \rangle$ in $\mathbf{B}(\bar{A}, \bar{X})$ w.r.t. a block term order such that $\bar{X} \gg \bar{A}$. In the latter method, we have to compute again a (stratified) Boolean Gröbner basis of $\langle F \rangle$ in a Boolean polynomial ring $(\mathbf{B}(\bar{A})/I)(\bar{X})$ in order to compute a (stratified) comprehensive Boolean Gröbner basis of F on I . Even if we can decrease consumed memory space, we need more computation time. Fortunately, however, a Boolean Gröbner basis computed with a block term order is already a comprehensive Boolean Gröbner basis of F on I . In order to prove this fact, we need the following well-known facts which are easy in themselves.

Lemma 20. *Let $R[\bar{A}, \bar{X}]$ be a polynomial ring with variables \bar{A} and \bar{X} over a commutative ring R with an identity. Let I be an ideal of this polynomial ring. Let $>$ be a term order of $T(\bar{X})$ and G be a Gröbner basis of I w.r.t. $>$ regarding $R[\bar{A}, \bar{X}]$ as a polynomial ring over the coefficient ring $R[\bar{A}]$, that is $\langle \{LM(g) | g \in G\} \rangle = \langle \{LM(f) | f \in I\} \rangle$. Let $I_{\bar{A}}$ be $I \cap R[\bar{A}]$. Let ψ be a natural homomorphism from $R[\bar{A}]$ to $R[\bar{A}]/I_{\bar{A}}$ induced by $I_{\bar{A}}$. Then $\psi(G \setminus G \cap R[\bar{A}])$ is a Gröbner basis of the ideal $\psi(I)$ in $(R[\bar{A}]/I_{\bar{A}})[\bar{X}]$.*

Lemma 21. *Let $R[\bar{A}, \bar{X}]$ and I be the same as the above lemma. Let $>$ be a block term order of $T(\bar{A}, \bar{X})$ such that $\bar{X} \gg \bar{A}$ and G be a Gröbner basis of I w.r.t. $>$. Then G is also a Gröbner basis of I w.r.t. $>_{\bar{X}}$ regarding $R[\bar{A}, \bar{X}]$ as a polynomial ring over the coefficient ring $R[\bar{A}]$, where $>_{\bar{X}}$ denotes a restriction of $>$ to $T(\bar{X})$.*

In the above lemmas, obviously we can replace R by a Boolean ring \mathbf{B} , furthermore the lemmas also hold if we replace $R[\bar{A}, \bar{X}]$ and $R[\bar{A}]$ by $\mathbf{B}(\bar{A}, \bar{X})$ and $\mathbf{B}(\bar{A})$ respectively. By this observation together with Theorem 16, the following our main theorem directly follows.

Theorem 22. *Let $G = \{g_1(\bar{A}, \bar{X}), \dots, g_k(\bar{A}, \bar{X})\}$ be a Boolean Gröbner basis of $F = \{f_1(\bar{A}, \bar{X}), \dots, f_l(\bar{A}, \bar{X})\}$ in a Boolean polynomial ring $\mathbf{B}(\bar{A}, \bar{X})$ w.r.t. a block term order $>$ such that $\bar{X} \gg \bar{A}$. Then G is a comprehensive Boolean Gröbner basis of F w.r.t. $>_{\bar{X}}$, furthermore $G \setminus G \cap \mathbf{B}(\bar{A})$ is a comprehensive Boolean Gröbner basis of F on $\langle F \rangle \cap \mathbf{B}(\bar{A})$ w.r.t. $>_{\bar{X}}$.*

In the above theorem, $G = \{g_1(\bar{a}, \bar{X}), \dots, g_k(\bar{a}, \bar{X})\}$ may not be stratified or reduced even if $G = \{g_1(\bar{A}, \bar{X}), \dots, g_k(\bar{A}, \bar{X})\}$ is stratified, because G may not be reduced as a Boolean Gröbner basis either in $(\mathbf{B}(\bar{A})/\langle F \rangle \cap \mathbf{B}(\bar{A}))(\bar{X})$ or $(\mathbf{B}(\bar{A}))(\bar{X})$. In order to have a stratified comprehensive Boolean Gröbner basis, we need further computation to transform G to be stratified in a Boolean polynomial ring $(\mathbf{B}(\bar{A})/\langle F \rangle \cap \mathbf{B}(\bar{A}))(\bar{X})$. If G is stratified in $\mathbf{B}(\bar{A}, \bar{X})$, each coefficient of members of G has a canonical form as an element of $\mathbf{B}(\bar{A})/\langle F \rangle \cap \mathbf{B}(\bar{A})$, i.e. a normal form by a Boolean Gröbner basis $G \cap \mathbf{B}(\bar{A})$. It is not a heavy computation to make G stratified using $G \cap \mathbf{B}(\bar{A})$ for the computation of the residue class ring $\mathbf{B}(\bar{A})/\langle F \rangle \cap \mathbf{B}(\bar{A})$.

Example 3. *The following left constraint is same as the previous example except that we have an another unknown variable a for an element. Using another*

set variable A to represent a singleton set $\{a\}$, it is equivalent to the right system of equations of a Boolean polynomial ring $\mathbf{B}(A, X, Y)$.

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

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

The stratified Boolean Gröbner basis G of F w.r.t. a lexicographic term order such that $X > Y > A$ has the following form:

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

This is not stratified or even reduced as a Boolean Gröbner basis in $(\mathbf{B}(A))(X, Y)$ or $(\mathbf{B}(A)/\langle(1 + \{s_2\})A\rangle)(X, Y)$. In fact, if we specialize A by $\{s_2\}$, $G(\{s_2\}) \setminus \{0\}$ becomes $\{\{s_2\}XY, \{s_2\}Y + \{s_2\}, (1 + \{s_2\})Y, \{s_2\}X, (1 + \{s_2\})X + \{s_1\}\}$, which is not stratified or even reduced.

The stratified Boolean Gröbner basis of F in $(\mathbf{B}(A)/\langle(1 + \{s_2\})A\rangle)(X, Y)$ has the following form:

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

Whereas the stratified Boolean Gröbner basis of F in $(\mathbf{B}(A))(X, Y)$ has the following form:

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

In the above example, the last two stratified Gröbner bases are not so different. If we have many parameters and the \bar{X} -eliminate portion $\langle F \rangle \cap \mathbf{B}(\bar{A})$ is not very simple, the stratified Gröbner basis in $(\mathbf{B}(\bar{A}))(\bar{X})$ is much bigger than the stratified Gröbner basis in $(\mathbf{B}(\bar{A})/\langle F \rangle \cap \mathbf{B}(\bar{A}))(\bar{X})$ in general because we do not use any simplification by $G \cap \mathbf{B}(\bar{A})$.

Let us conclude this section with the following obvious but important fact, which actually plays an important role in our recent work [11] of a non-trivial application of Boolean Gröbner bases.

Corollary 23. *Let $G = \{g_1(\bar{X}), \dots, g_k(\bar{X})\}$ be a Boolean Gröbner basis of $F = \{f_1(\bar{X}), \dots, f_l(\bar{X})\}$ in a Boolean polynomial ring $\mathbf{B}(\bar{X})$ w.r.t. a purely lexicographic term order $>$ such that $\bar{X}_n > X_{n-1} > \dots > X_1$. Then G is a Boolean comprehensive Gröbner basis of F regarding X_i, \dots, X_1 as parameters, for each $i = 1, \dots, n - 1$.*

5 Implementation

The most important merit of Theorem 22 is that we can construct a comprehensive Boolean Gröbner basis by computing only usual Gröbner bases in a

polynomial ring over \mathbb{GF}_2 with the algorithm described in section 3 in case we have an isomorphism $\mathbf{B}_F \simeq \mathbb{GF}_2^k$ for relatively small k , thereby we can easily implement computation of comprehensive Boolean Gröbner bases in any computer algebra system with a facility to compute Gröbner bases in polynomial rings over \mathbb{GF}_2 . For the case that \mathbf{B} is a set of all finite or co-finite subsets of S , where S is a set of all strings, we implemented our method described in section 3 and 4 in a computer algebra system Risa/Asir.

For a given 9×9 Sudoku puzzle, if we associate a variable X_{ij} for each grid at the i -th row and the j -th column, it can be considered as a set constraint where each variable should be assigned a singleton set from 9 candidates $\{1\}, \{2\}, \dots, \{9\}$ so that any distinct two variables which lie on a same row, column or block must be assigned different singleton sets. As initial conditions, several variables (17 to 25 in mosts high level puzzles) are assigned singleton sets. This constraint is translated into a system of equations of a Boolean polynomial ring $\mathbf{B}(X_{11}, X_{12}, \dots, X_{99})$ with 81 variables as follows:

- (1) Initial conditions such as $X_{11} = \{4\}, X_{15} = \{9\}, \dots$
- (2) $X_{ij}X_{i'j'} = 0 (= \emptyset)$ for each pair of distinct variables $X_{ij}, X_{i'j'}$ which lie on a same row, column or block.
- (3) $\sum_{(i,j) \in A} X_{ij} = 1 (= \{1, 2, \dots, 9\})$ where A is a set of indices lying on a same row, column or block. (There are 27 such A 's.)

We applied our implementation to solve such equations in [11]. The following table includes data of our computations of Boolean Gröbner bases w.r.t. a purely lexicographic term order such that $X_{99} > \dots > X_{91} > \dots > X_{11}$ for solving 12 Sudoku puzzles each of which has 17 initial conditions and ranked as extremely difficult (thereby we have 64 unknown variables to solve). Each Boolean Gröbner basis can be considered as a simultaneous comprehensive Boolean Gröbner basis by Corollary 23, which is especially important for our application. We used a linux machine with 2GB memory and CPU Core2Duo 2GHZ. The data contains computation time (in terms of second) by our implementation (namely Asir) and old implementation of [9] (namely Klic0). We also include computation time of comprehensive Boolean Gröbner bases with old implementation of [9] which is based on the method described in section 2.3 in the column Klic1 and Klic2, where X_{11} is treated as a parameter in Klic1 and X_{11}, X_{12} in Klic2. The symbol ∞ means that the computation did not terminate within 2 hours.

puzzle	Asir	Klic0	Klic1	Klic2
1	41.7	85.3	134.1	592.3
2	135.5	∞	∞	∞
3	48.1	∞	∞	∞
4	83.9	∞	∞	∞
5	44.3	1242.3	2657.2	2309.1
6	76.2	∞	∞	∞

computation data 1

puzzle	Asir	Klic0	Klic1	Klic2
7	43.6	398.3	643.3	850.6
8	80.4	∞	∞	∞
9	40.1	1025.3	1179.4	1089.3
10	85.8	∞	∞	∞
11	92.6	∞	∞	∞
12	48.9	686.5	1233.1	1634.7

computation data 2

From this rather small data, we can see remarkable speedup of our new implementation.

6 Conclusions

Essentially the same assertion of Theorem 22 has been reported in [10]. The proof we gave there is based on a fact concerning stability of a specialization with a zero of a zero-dimensional ideal. It is not only complicated but also turned out to be incomplete. The proof we give in this paper does not employ any tricky technique but is a simple and natural consequence if we are sufficiently familiar with structures of Boolean polynomial rings.

References

1. Kapur, D.: An Approach for Solving Systems of Parametric Polynomial Equations. In: Saraswat, Van Hentenryck (eds.) Principles and Practices of Constraint Programming, pp. 217–244. MIT Press, Cambridge (1995)
2. Manubens, M., Montes, A.: Improving DISPGB algorithm using the discriminant ideal. *J. Symb. Comp.* 41, 1245–1263 (2006)
3. Montes, A.: A new algorithm for discussing Gröbner bases with parameters. *J. Symb. Comp.* 33(2), 183–208 (2002)
4. Noro, M., et al.: A Computer Algebra System Risa/Asir (2009), <http://www.math.kobe-u.ac.jp/Asir/asir.html>
5. Sakai, K., Sato, Y.: Boolean Gröbner bases. ICOT Technical Memorandum 488 (1988), <http://www.icot.or.jp/ARCHIVE/Museum/TRTM/tm-list-E.html>
6. Sakai, K., Sato, Y., Menju, S.: Boolean Gröbner bases(revised). ICOT Technical Report 613. (1991), <http://www.icot.or.jp/ARCHIVE/Museum/TRTM/tr-list-E.html>
7. Sato, Y., et al.: Set Constrains Solvers(Prolog version) (1996), <http://www.icot.or.jp/ARCHIVE/Museum/FUNDING/funding-96-E.html>
8. Sato, Y.: A new type of canonical Gröbner bases in polynomial rings over Von Neumann regular rings. In: Proceedings of ISSAC 1998, pp. 317–332. ACM Press, New York (1998)
9. Sato, Y., et al.: Set Constrains Solvers(Klic version) (1998), <http://www.icot.or.jp/ARCHIVE/Museum/FUNDING/funding-98-E.html>
10. Sato, Y., Inoue, S.: On the Construction of Comprehensive Boolean Gröbner Bases. In: Proceedings of the Seventh Asian Symposium on Computer Mathematics (ASCM 2005), pp. 145–148 (2005)
11. Sato, Y., Inoue, S., Suzuki, A., Nabeshima, K.: Boolean Gröbner Bases and Sudoku (submitted for publication)
12. Sato, Y., Nagai, A., Inoue, S.: On the Computation of Elimination Ideals of Boolean Polynomial Rings. In: Kapur, D. (ed.) ASCM 2007. LNCS (LNAI), vol. 5081, pp. 334–348. Springer, Heidelberg (2008)
13. Suzuki, A., Sato, Y.: An Alternative approach to Comprehensive Gröbner Bases. *J. Symb. Comp.* 36(3-4), 649–667 (2003)
14. Suzuki, A., Sato, Y.: A Simple Algorithm to Compute Comprehensive Gröbner Bases Using Gröbner Bases. In: International Symposium on Symbolic and Algebraic Computation (ISSAC 2006), Proceedings, pp. 326–331 (2006)
15. Weispfenning, V.: Gröbner bases in polynomial ideals over commutative regular rings. In: Davenport, J.H. (ed.) ISSAC 1987 and EUROCAL 1987. LNCS, vol. 378, pp. 336–347. Springer, Heidelberg (1989)
16. Weispfenning, V.: Comprehensive Gröbner bases. *J. Symb. Comp.* 14(1), 1–29 (1992)

On Invariant Manifolds of Dynamical Systems in Lie Algebras

Valentin Irtegov and Tatyana Titorenko

Institute for System Dynamics and Control Theory SB RAS,
134, Lermontov str., Irkutsk, 664033, Russia
irtegov@icc.ru

Abstract. Some problems of obtaining, analysis of stability and bifurcations of invariant sets of dynamical systems described by Euler equations in Lie algebras $so(4)$ and $so(3,1)$ are discussed. The considered systems assume additional polynomial first integrals of the 3rd and 6th degrees. Invariant sets of these systems can be found from the conditions of stationarity for the problem first integrals. Methods of computer algebra have been employed in the capacity of the computational methods. The computer algebra systems (CAS) *Mathematica* and *Maple* have been used.

1 Introduction

In recent years, there appeared a series of publications devoted to finding completely integrable cases of Euler equations in Lie algebras [1]–[3]. Such equations often have a rather good mechanical interpretation, for example, the generalized Kirchhoff equations, the Poincaré–Zhukowsky equations [4], the Euler–Arnold tops in algebra $sl(2, \mathbb{C})$ as two-particle systems like the Cologero systems [5]. In the present paper, for the two systems of equations of above type [6], [7], where additional algebraic homogeneous integrals have the degree of 3 and 6, we have considered the problem of finding and investigation of invariant sets, on which the elements of the family of the problem’s first integrals assume a stationary value. We call such sets the invariant manifolds of steady motions (IMSMs). The class of such IMSMs is interesting for the fact that Lyapunov’s 2nd method is more suitable for investigation of their stability. To the end of finding invariant manifolds we use the family of the problem basic first integrals. It allows one to obtain the families of IMSMs. We have considered problems of bifurcation and stability in the sense of Lyapunov for a series of the families of IMSMs obtained.

The problem of obtaining “resonance” invariant manifolds, on which the basic first integrals are related by some polynomial relationships, has also been considered.

We used CAS *Mathematica* and *Maple* for performing necessary computations. All the computations have been conducted on a computer having the processor Intel Core 2 Duo (2.4 GHz) and 2 GB RAM running under Windows XP.

2 Euler Equations on $so(4)$

2.1 Finding Invariant Manifolds

Consider a dynamical system described by the differential equations:

$$\begin{aligned}
 \dot{M}_1 &= 2M_2(\alpha_1 M_1 + 2\alpha_3 M_3) - (M_3 \gamma_1 - M_1 \gamma_3), \\
 \dot{M}_2 &= 2(\alpha_1 M_3 - \alpha_3 M_1)M_3 - 2(\alpha_1 M_1 + \alpha_3 M_3)M_1 + (M_2 \gamma_3 - M_3 \gamma_2), \\
 \dot{M}_3 &= -2\alpha_1 M_2 M_3, \\
 \dot{\gamma}_1 &= 2(\alpha_1 M_1 + \alpha_3 M_3)\gamma_2 + (2\alpha_3 M_2 - \gamma_1)\gamma_3 + kM_1 M_3, \\
 \dot{\gamma}_2 &= 2(\alpha_1 M_3 - \alpha_3 M_1)\gamma_3 - 2(\alpha_1 M_1 + \alpha_3 M_3)\gamma_1 + kM_2 M_3 - \gamma_2 \gamma_3, \\
 \dot{\gamma}_3 &= -2(\alpha_1 M_3 - \alpha_3 M_1)\gamma_2 - k(M_1^2 + M_2^2) - 2\alpha_3 M_2 \gamma_1 + \gamma_1^2 + \gamma_2^2, \tag{1}
 \end{aligned}$$

where M_i, γ_i ($i = 1, 2, 3$) are the components of the two 3-dimensional vectors, α_1, α_2 are some constants, $k = -\alpha_1^2 - \alpha_2^2$.

Equations (I) have the first integrals:

$$\begin{aligned}
 V_0 &= \alpha_3(M_1^2 + M_2^2 + M_3^2) - 2M_3(\alpha_1 M_1 + \alpha_3 M_3) + M_1 \gamma_2 - M_2 \gamma_1 = h, \\
 V_1 &= M_1 \gamma_1 + M_2 \gamma_2 + M_3 \gamma_3 = c_1, \quad V_2 = k(M_1^2 + M_2^2 + M_3^2) + \gamma_1^2 + \gamma_2^2 + \gamma_3^2 = c_2, \\
 V_3 &= \{2[\alpha_3(M_1 \gamma_2 - M_2 \gamma_1) + \alpha_1(M_2 \gamma_3 - M_3 \gamma_2)] - k(M_1^2 + M_2^2 + M_3^2) \\
 &\quad + \gamma_1^2 + \gamma_2^2 + \gamma_3^2\} M_3 = c_3 \quad (h, c_i = const). \tag{2}
 \end{aligned}$$

Here V_3 is the additional cubic integral found in [6]. In this paper, one of applications for the systems similar to (I) is given. Such systems may also represent the interest, for example, within the framework of the generalized Kirchhoff model, which describes motion of a rigid body in fluid, or the Poincaré model, which describes motion of a rigid body with a cavity filled with ideal rotational fluid [4].

Let us consider the problem of obtaining invariant manifolds for equations (I). To this end, we shall apply the Routh–Lyapunov method and its modifications [8]–[10].

According to the modified Routh–Lyapunov method, invariant manifolds of differential equations (I) may be found by solving the problem of conditional extremum for the first integrals of the problem. For this purpose, some combinations of the problem first integrals (families of first integrals) are formed. We restrict our consideration to the linear combinations of first integrals:

$$K = \lambda_0 V_0 - \lambda_1 V_1 - \frac{\lambda_2}{2} V_2 - \lambda_3 V_3 \quad (\lambda_i = const). \tag{3}$$

Here $\lambda_0, \lambda_1, \lambda_2, \lambda_3$ are some constants, which may assume also zero values.

Next, while following the technique chosen, write stationary conditions for the integral K with respect to the variables $M_1, M_2, M_3, \gamma_1, \gamma_2, \gamma_3$:

$$\begin{aligned}
 \lambda_0 \gamma_2 - \lambda_1 \gamma_1 + (2\alpha_3 \lambda_0 - k \lambda_2) M_1 - 2\alpha_1 \lambda_0 M_3 + 2\lambda_3 (k M_1 - \alpha_3 \gamma_2) M_3 &= 0, \\
 \lambda_0 \gamma_1 + \lambda_1 \gamma_2 - (2\alpha_3 \lambda_0 - k \lambda_2) M_2 + 2\lambda_3 [\alpha_1 \gamma_3 - \alpha_3 \gamma_1 - k M_2] M_3 &= 0,
 \end{aligned}$$

$$\begin{aligned}
& \lambda_1 \gamma_3 + 2\alpha_1 \lambda_0 M_1 + (2\alpha_3 \lambda_0 + k\lambda_2)M_3 - \lambda_3[k(M_1^2 + M_2^2 + 3M_3^2) \\
& \quad - 2(\alpha_1 \gamma_3 - \alpha_3 \gamma_1)M_2 - 2(\alpha_3 M_1 - 2\alpha_1 M_3)\gamma_2 - (\gamma_1^2 + \gamma_2^2 + \gamma_3^2)] = 0, \\
& \lambda_1 M_1 + \lambda_0 M_2 + \lambda_2 \gamma_1 - 2\lambda_3(\alpha_3 M_2 - \gamma_1)M_3 = 0, \\
& \lambda_0 M_1 - \lambda_1 M_2 - \lambda_2 \gamma_2 + 2\lambda_3[(\alpha_1 M_3 - \alpha_3 M_1) - \gamma_2]M_3 = 0, \\
& \lambda_1 M_3 + \lambda_2 \gamma_3 + 2\lambda_3(\alpha_1 M_2 + \gamma_3)M_3 = 0.
\end{aligned} \tag{4}$$

In the case under consideration, equations (4) represent a system of the 2nd degree algebraic equations. The parameters here are λ_i, α_j .

The solutions of equations (4) allow one to define both the families of stationary solutions and the families of IMSMs of differential equations (1), which correspond to the family of first integrals K . We take interest in solutions of system (1), which represent families of IMSMs. To this end, we have to consider the cases, when equations (4) are dependent. The conditions of existence of such degenerations of system (4) may be found by equating the system Jacobian to zero. Solutions of equations (4), which have been obtained under the conditions indicated, are, generally speaking, the desired IMSMs. Computation of the Jacobian of the parametric polynomial system of equations and its analysis may represent a rather complex problem because of the system dimension and the bulky character of expressions for the coefficients entering in it, etc. We employed computer algebra methods for obtaining the solutions of interest for us.

Let us conduct a preliminary qualitative analysis of the solution set of equations (4), while using the *Maple* program package “PolynomialIdeals”. The programs of this package allow one, for example, to obtain the answer to the question whether the system under investigation has finite or infinite number of solutions, to find the dimension of its solution set, if the number of solutions is infinite, or to determine the number of solutions for the system having a finite number of solutions, etc. In particular, we have used the programs “IsZeroDimensional”, “NumberOfSolutions”, “HilbertDimension” and some other programs.

Analysis for the solution set of equations (4) was conducted over the field $Q(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \alpha_1, \alpha_2) [M_1, M_2, M_3, \gamma_1, \gamma_2, \gamma_3]$. The following results have been obtained. The system has zero-dimensional sets (stationary solutions) in the capacity of its solutions, and their number is finite. On the whole, there are 6 solutions. In given case, by computing the Gröbner lexicographic basis it is easy to verify that system (4) over the indicated field has indeed 6 solutions of above type (including the zero one).

Equations (4) may have nonzero-dimensional sets in the capacity of their solutions in the phase space, when some part of the parameters, for example, one of λ_i , are included into the number of variables. With the aid of the program *Mathematica* “GroebnerBasis” we have constructed the lexicographic bases with respect to the variables $\gamma_1, \gamma_2, \gamma_3, M_1, M_2, M_3, \lambda_i$. All the bases constructed contain the equation:

$$\lambda_1^2 + (\lambda_0 + \alpha_3 \lambda_2)^2 + \alpha_1^2 \lambda_2^2 = 0. \tag{5}$$

Equation (5) may be considered as the condition imposed on the parameters λ_i, α_j . If this condition is satisfied, then system (4) has the solutions of dimension higher than zero.

Real solutions for the quadratic equation (5) with respect to λ_0 are:

$$i) \lambda_0 = -\alpha_3\lambda_2, \lambda_1 = 0, \alpha_1 = 0; \quad ii) \lambda_0 = \lambda_1 = \lambda_2 = 0.$$

Under the above conditions imposed on the parameters λ_i, α_j we have found the following solutions of the stationary equations:

$$\begin{aligned} \gamma_1 &= \alpha_3M_2, \gamma_2 = -\alpha_3M_1, \gamma_3 = 0, M_3 = \frac{\lambda_2}{\lambda_3} \text{ when } \lambda_0 = -\alpha_3\lambda_2, \lambda_1 = 0, \alpha_1 = 0; \\ \gamma_1 &= \alpha_3M_2, \gamma_2 = -\alpha_3M_1, \gamma_3 = 0, M_3 = 0 \text{ when } \lambda_0 = -\alpha_3\lambda_2, \lambda_1 = 0, \alpha_1 = 0; \\ \gamma_1 &= \alpha_3M_2, \gamma_2 = -\frac{\alpha_1^2 + \alpha_3^2}{\alpha_3}M_1, \gamma_3 = -\alpha_1M_2, M_3 = -\frac{\alpha_1}{\alpha_3}M_1 \text{ when } \lambda_0 = \lambda_1 \\ &= \lambda_2 = 0; \\ \gamma_1 &= \alpha_3M_2, \gamma_2 = 0, \gamma_3 = -\alpha_1M_2, M_1 = 0, M_3 = 0 \text{ when } \lambda_0 = \lambda_1 = \lambda_2 = 0. \end{aligned} \quad (6)$$

Analysis of solutions (6) has given evidence that these represent families of IMSMs for the differential equations (II).

When solving the stationary equations with respect to some part of the parameters λ_i and the phase variables, for example, $M_2, \gamma_1, \gamma_2, \gamma_3, \lambda_0, \lambda_1$, it is possible to obtain also several families of IMSMs. Using the technique of Gröbner bases, we have found the following solutions of the stationary equations with respect to the indicated variables:

$$\begin{aligned} \gamma_1 &= \pm \frac{\sqrt{M_1z_1}}{M_3}, \quad \gamma_2 = -2\left[\frac{\alpha_1M_1^2}{M_3} - \alpha_1M_3 + 2\alpha_3M_1\right], \quad \gamma_3 = \pm \frac{\sqrt{z_1}}{\sqrt{M_1}}, \quad M_2 = 0, \\ \lambda_0 &= \frac{2\alpha_1[\lambda_2(M_3^2 - M_1^2) + \lambda_3(M_3^2 - 2M_1^2)M_3]}{M_1M_3} - 2\alpha_3(3\lambda_3M_3 + 2\lambda_2), \\ \lambda_1 &= \mp \frac{\sqrt{M_1z_1}(2\lambda_3M_3 + \lambda_2)}{\sqrt{M_1}M_3}; \end{aligned} \quad (7)$$

$$\begin{aligned} \gamma_1 &= \frac{2\lambda_3[\alpha_1\lambda_3M_1 + \alpha_3(2\lambda_3M_3 + \lambda_2)]M_2M_3 \pm M_1\sqrt{z_2}}{z_3}, \\ \gamma_2 &= \frac{2\lambda_3[\alpha_1(\lambda_2M_3 + \lambda_3(M_3^2 - M_1^2)) - \alpha_3(2\lambda_3M_3 + \lambda_2)M_1]M_3 \pm M_2\sqrt{z_2}}{z_3}, \\ \gamma_3 &= \frac{M_3[-2\alpha_1\lambda_3M_2(\lambda_3M_3 + \lambda_2) \pm \sqrt{z_2}]}{z_3}, \\ \lambda_0 &= -\frac{2\lambda_3^2M_3(\alpha_1M_1 + \alpha_3M_3)}{\lambda_3M_3 + \lambda_2}, \quad \lambda_1 = \mp \frac{z_2}{\lambda_3M_3}. \end{aligned} \quad (8)$$

Here $z_1 = \alpha_1(3\alpha_1M_1 + 2\alpha_3M_3)M_3^2 - (2\alpha_1M_1 + 3\alpha_3M_3)^2M_1$,
 $z_2 = \alpha_3(\lambda_2 - \lambda_3M_3)[4\alpha_1\lambda_3(2\lambda_3M_3 + \lambda_2)M_1M_3 + \alpha_3^2(\lambda_3M_3 - \lambda_2)(2\lambda_3M_3 + \lambda_2)^2]$
 $- \alpha_1^2[\lambda_3^2(\lambda_3M_3 + \lambda_2)^2 + 4\lambda_3^4M_1^2M_3^2]$, $z_3 = (\lambda_3M_3 + \lambda_2)(2\lambda_3M_3 + \lambda_2)$.

The first four equations (7) define the family of IMSMs for the differential equations (II) written in different maps. The vector field on the elements of the given family of IMSMs is described by the equations:

$$\dot{M}_1 = 0, \quad \dot{M}_3 = 0. \quad (9)$$

The latter two equations (7) for λ_0 and λ_1 represent the first integrals of differential equations (9). In the given case, these integrals shall obviously be trivial due to the fact that the equations themselves are trivial.

In (8), the first three equations define the family of IMSMs for the differential equations (1), which is also written in different maps. The vector field on the elements of the given family is written by the equations:

$$\begin{aligned} \dot{M}_1 &= \frac{2M_2[\alpha_1\lambda_2M_1 + \alpha_3M_3(\lambda_3M_3 + 2\lambda_2)]}{\lambda_3M_3 + \lambda_2}, \\ \dot{M}_2 &= -\frac{2}{z_3}[\alpha_3(\lambda_2(2\lambda_3M_3 + \lambda_2) + z_3)M_1M_3 + \alpha_1(\lambda_2^2(M_1^2 - M_3^2) \\ &\quad + \lambda_2\lambda_3(3M_1^2 + M_2^2 - 2M_3^2)M_3 + \lambda_3^2(M_1^2 + M_2^2 - M_3^3)M_3^2)], \\ \dot{M}_3 &= -2\alpha_1M_2M_3. \end{aligned} \tag{10}$$

The latter two equations (8) represent the first integrals of differential equations (10).

2.2 On Bifurcations of Invariant Manifolds

Consider the problem of branching for above IMSMs. For the purpose of simplicity we restrict our consideration with trivial conditions imposed on the problem's parameters.

It can easily be shown that all the families of IMSMs (6) adjunct to the 0th solution. Indeed, compute the Jacobian of system (4) for $M_i = 0$, $\gamma_i = 0$ ($i = 1, 2, 3$). The expression obtained for the Jacobian writes:

$$\begin{aligned} J &= (\lambda_1^2 + (\lambda_0 + \alpha_3\lambda_2)^2 + \alpha_1^2\lambda_2^2)[\lambda_0^2(\lambda_1^2 - 2\alpha_3\lambda_0\lambda_2 - 3(\alpha_1^2 + \alpha_3^2)\lambda_2^2) \\ &\quad + (\lambda_1^2 + (\alpha_1^2 + \alpha_3^2)\lambda_2^2)^2]. \end{aligned}$$

Having assumed that $J = 0$, find the conditions under which the families of IMSMs, that adjunct to the 0th solution, can be found as the solutions of stationary equations (4). Such conditions, in particular, are:

$$(i) \lambda_0 = -\alpha_3\lambda_2, \lambda_1 = 0, \alpha_1 = 0; \quad (ii) \lambda_0 = \lambda_1 = \lambda_2 = 0. \tag{11}$$

When comparing the restrictions imposed on λ_i , for which the families of IMSMs (6) have been obtained, with (11), we can conclude that these families of IMSMs adjunct to the 0th solution.

By analogy, we can find the conditions and the families of IMSMs (that correspond to these conditions), which adjunct to families of IMSMs (7) and (8). Such conditions for (7) are (a) $\alpha_1 = 0$, (b) $\lambda_3 = 0$. In case of (8), such condition is $\lambda_3 = 0$.

It follows from the condition $\alpha_1 = 0$ that the first two families of IMSMs (6) adjunct to the family of IMSMs (7). In case of $\lambda_3 = 0$, we have found one-dimensional and two-dimensional families of IMSMs, which adjunct both to the family of IMSMs (7) and to (8). One of these families may be obtained, while assuming $\lambda_3 = 0$ in (7).

3 Euler Equations on $\mathfrak{so}(3,1)$

3.1 Obtaining Invariant Manifolds

Consider a dynamical system described in [7]. Its differential equations write:

$$\begin{aligned} \dot{M}_1 &= 2(\gamma_1 M_3 - \gamma_3 M_1) - M_2(\alpha_1 M_1 + 2\alpha_3 M_3), \\ \dot{M}_2 &= 2(\gamma_3 M_2 - \gamma_2 M_3) + [M_1(\alpha_1 M_1 + \alpha_3 M_3) - M_3(\alpha_1 M_3 - \alpha_3 M_1)], \\ \dot{M}_3 &= \alpha_1 M_2 M_3, \\ \dot{\gamma}_1 &= 2[\alpha_3(\gamma_2 M_3 - \gamma_3 M_2) - (\alpha_3 M_2 - \gamma_1)\gamma_3] - (\alpha_1 \gamma_2 + 2k M_3) M_1, \\ \dot{\gamma}_2 &= \alpha_1(\gamma_1 M_1 - \gamma_3 M_3) + 2[\alpha_3(\gamma_3 M_1 - \gamma_1 M_3) + (\alpha_3 M_1 + \gamma_2)\gamma_3 - k M_2 M_3], \\ \dot{\gamma}_3 &= 4\alpha_3(\gamma_1 M_2 - \gamma_2 M_1) + \alpha_1 \gamma_2 M_3 + 2[k(M_1^2 + M_2^2) - (\gamma_1^2 + \gamma_2^2)]. \end{aligned} \tag{12}$$

Equations (12) have the following first integrals:

$$\begin{aligned} 2\bar{V}_0 &= 2(\gamma_2 M_1 - \gamma_1 M_2) - M_3(\alpha_1 M_1 + \alpha_3 M_3) + 2\alpha_3(M_1^2 + M_2^2 + M_3^2) = 2\bar{h}, \\ \bar{V}_1 &= \gamma_1 M_1 + \gamma_2 M_2 + \gamma_3 M_3 = \bar{c}_1, \quad \bar{V}_2 = k(M_1^2 + M_2^2 + M_3^2) + \gamma_1^2 + \gamma_2^2 + \gamma_3^2 = \bar{c}_2, \\ \bar{V}_3 &= \left\{ (\alpha_3 M_1 + \gamma_2)^2 M_1^2 + [(\alpha_3 M_2 - \gamma_1) M_1 - (\alpha_1 M_2 + \gamma_3) M_3]^2 \right. \\ &\quad \left. - 2\alpha_1(\alpha_3 M_1 + \gamma_2)(M_1^2 + M_3^2) M_3 + [(\alpha_3 M_1 + \gamma_2)\gamma_2 + (\alpha_3 \gamma_2 - k M_1) M_1 \right. \\ &\quad \left. + \alpha_1^2 M_3^2] M_3^2 \right\} M_3^2 = \bar{c}_3 \quad (\bar{h}, \bar{c}_i = const). \end{aligned} \tag{13}$$

Here $M_i, \gamma_i, \lambda_j, \alpha_l, k$ have the same sense as in section 2.1. \bar{V}_3 is the additional integral of the 6th degree obtained in paper [7].

Let us consider the problem of obtaining invariant manifolds for equations (12). For this purpose, we introduce the function

$$\bar{K} = 2\lambda_0 \bar{V}_0 - \lambda_1 \bar{V}_1 - \frac{\lambda_2}{2} \bar{V}_2 - \frac{\lambda_3}{2} \bar{V}_3 \quad (\lambda_i = const)$$

and write down the stationary conditions for \bar{K} with respect to the variables $M_1, M_2, M_3, \gamma_1, \gamma_2, \gamma_3$

$$\begin{aligned} &\lambda_0(2(\alpha_3 M_1 + \eta) - \alpha_1 M_3) - \lambda_1 \gamma_1 - \lambda_2 k M_1 + \lambda_3[\alpha_3(\alpha_1 M_3 - \eta) M_1^2 \\ &\quad - (\eta^2 + \nu^2 - (2\alpha_1 \eta + k M_3) M_3) M_1 + (\nu \rho + \alpha_3 M_3 \sigma) M_3] M_3^2 = 0, \\ &2\lambda_0(\alpha_3 M_2 + \nu) - \lambda_1 \gamma_2 - \lambda_2 k M_2 - \lambda_3(\alpha_1 M_3 - \alpha_3 M_1)(\rho M_3 - \nu M_1) M_3^2 = 0, \\ &(2\alpha_3 M_3 - \alpha_1 M_1) \lambda_0 - \lambda_1 \gamma_3 - \lambda_2 k M_3 - \lambda_3[(\eta^2 + \nu^2) M_1^2 + \alpha_1(2\alpha_1 M_3 - 3\eta) \\ &\quad \times (M_1^2 + M_3^2) M_3 + (2(\eta^2 + \rho^2) + \alpha_1 M_3(\alpha_1 M_3 - 2\eta)) M_3^2 - 3\nu \rho M_1 M_3] M_3 = 0, \\ &2\lambda_0 M_2 + \lambda_1 M_1 + \lambda_2 \gamma_1 + \lambda_3(\rho M_3 - \nu M_1) M_1 M_3^2 = 0, \\ &2\lambda_0 M_1 - \lambda_1 M_2 - \lambda_2 \gamma_2 + \lambda_3[(\alpha_1 M_3 - \eta) M_1^2 + (\sigma - \alpha_3 M_1) M_3^2] M_3^2 = 0, \\ &\lambda_1 M_3 + \gamma_3 \lambda_2 + \lambda_3(\rho M_3 - \nu M_1) M_3^3 = 0, \end{aligned} \tag{14}$$

while using the following denotations $\eta = \alpha_3 M_1 + \gamma_2, \nu = \alpha_3 M_2 - \gamma_1, \rho = \alpha_1 M_2 + \gamma_3, \sigma = \alpha_1 M_3 - \gamma_2$.

The stationary equations obtained represent the system of polynomial algebraic equations of the 5th degree with the coefficients dependent on the parameters λ_i, α_j . Let us conduct analysis of their solution set, while using programs of the Maple package “PolynomialIdeals”.

In order to simplify the problem we have constructed the Gröbner basis of the scrutinized system with respect to the elimination monomial ordering. As a result, the initial system has decomposed into the two subsystems.

$$\begin{aligned}
 f_1(M_1, M_3, \lambda_i, \alpha_i) &= 0, & f_5(\gamma_3, M_1, M_2, M_3, \lambda_i, \alpha_i) &= 0, \\
 f_2(M_1, M_2, M_3, \lambda_i, \alpha_i) &= 0, & f_6(\gamma_2, \gamma_3, M_1, M_2, M_3, \lambda_i, \alpha_i) &= 0, \\
 f_3(\gamma_3, M_1, M_2, M_3, \lambda_i, \alpha_i) &= 0, & f_7(\gamma_1, \gamma_3, M_1, M_2, M_3, \lambda_i, \alpha_i) &= 0. \\
 f_4(\gamma_3, M_1, M_2, M_3, \lambda_i, \alpha_i) &= 0, & &
 \end{aligned} \tag{15}$$

$$\begin{aligned}
 M_2 &= 0, & g_4(M_1, M_3, \lambda_i, \alpha_i) &= 0, & g_8(M_1, M_3, \lambda_i, \alpha_i) &= 0, \\
 g_1(\gamma_1, M_1, M_3, \lambda_i, \alpha_i) &= 0, & g_5(M_1, M_3, \lambda_i, \alpha_i) &= 0, & g_9(M_1, M_3, \lambda_i, \alpha_i) &= 0, \\
 g_2(\gamma_2, M_1, M_3, \lambda_i, \alpha_i) &= 0, & g_6(M_1, M_3, \lambda_i, \alpha_i) &= 0, & g_{10}(M_1, M_3, \lambda_i, \alpha_i) &= 0. \\
 g_3(\gamma_3, M_1, M_3, \lambda_i, \alpha_i) &= 0, & g_7(M_1, M_3, \lambda_i, \alpha_i) &= 0, & &
 \end{aligned} \tag{16}$$

It is possible to investigate each of these subsystems separately. Here f_i, g_j are some polynomials of M_i, γ_i of degree of 5 to 9.

As a result of the analysis of the subsystems we have the following.

Equations (15) have two one-dimensional sets in the capacity of their solutions. By direct computation it is possible to obtain the sets. These represent families of IMSMs of differential equations (12). Equations of the 1st family of IMSMs write:

$$\begin{aligned}
 &(a_0M_3^{16} + a_1M_3^{14} + a_2M_3^{12} + a_3M_3^{10} + a_4M_3^8 + a_5M_3^6 + a_6M_3^4 + a_7M_3^2 \\
 &\quad + a_8)M_3^2 + (a_9M_3^{12} + a_{10}M_3^8 + a_{11}M_3^6 + a_{12}M_3^4 + a_{13}M_3^2 + a_{14})M_3^2\gamma_1^2 \\
 &\quad + (a_{15}M_3^{10} + a_{16}M_3^8 + a_{17}M_3^6 + a_{18}M_3^4 + a_{19}M_3^2 + a_{20})M_3\gamma_1 + a_{21} = 0, \\
 &(b_0M_3^{14} + b_1M_3^{12} + b_2M_3^{10} + b_3M_3^8 + b_4M_3^6 + b_5M_3^4 + b_6M_3^2 + b_7)M_3^2 + (b_8M_3^{12} \\
 &\quad + b_9M_3^8 + b_{10}M_3^6 + b_{11}M_3^4 + b_{12}M_3^2 + b_{13})M_3^2\gamma_2^2 + (b_{14}M_3^{14} + b_{15}M_3^{12} \\
 &\quad + b_{16}M_3^{10} + b_{17}M_3^8 + b_{18}M_3^6 + b_{19}M_3^4 + b_{20}M_3^2 + b_{21})M_3\gamma_2 + b_{22} = 0, \\
 &(c_0M_3^{14} + c_1M_3^{12} + c_2M_3^{10} + c_3M_3^8 + c_4M_3^6 + c_5M_3^4 + c_6M_3^2 + c_7)M_3^2 \\
 &\quad + (c_8M_3^{10} + c_9M_3^6 + c_{10}M_3^4 + c_{11}M_3^2 + c_{12})M_3^2\gamma_3^2 + (c_{13}M_3^8 + c_{14}M_3^6 \\
 &\quad + c_{15}M_3^4 + c_{16}M_3^2 + c_{17})M_3\gamma_3 + c_{18}\gamma_3^2 = 0, \\
 &\bar{a}_0M_3^2 + \bar{a}_1M_1M_3 + \bar{a}_2 = 0, \quad (\bar{b}_0M_3^4 + \bar{b}_1M_3^2 + \bar{b}_2M_2^2 + \bar{b}_3)M_3^2 + \bar{b}_4 = 0. \tag{17}
 \end{aligned}$$

Here $a_i, b_j, c_l, \bar{a}_m, \bar{b}_r$ are some expressions depending on $\alpha_1, \alpha_3, \lambda_0, \lambda_1, \lambda_2, \lambda_3$.

The vector field on the elements of the family of IMSMs (17) is defined by the equation:

$$\dot{M}_3 = \frac{\sqrt{p_0M_3^6 + p_1M_3^4 + p_2M_3^2 + p_3}}{2\alpha_1\lambda_2\sqrt{-\lambda_2\lambda_3z}}, \quad \text{where} \tag{18}$$

$$p_0 = -(\alpha_1^3 \lambda_0 \sqrt{-\lambda_2 \lambda_3} \lambda_3 (4\lambda_0^2 + 4\alpha_3 \lambda_0 \lambda_2 - k\lambda_2^2)z), \quad p_1 = \alpha_1^2 \lambda_2 \lambda_3 (8\lambda_0^2 + 6\alpha_3 \lambda_0 \lambda_2 - k\lambda_2^2)z^2, \quad p_2 = \alpha_1 \sqrt{-\lambda_2 \lambda_3} \lambda_2 (5\lambda_0 (4\lambda_0^2 + \lambda_1^2) + 2\alpha_3 (14\lambda_0^2 + \lambda_1^2) \lambda_2 + (4\alpha_1^2 + 13\alpha_3^2) \lambda_0 \lambda_2^2 - 2\alpha_3 k \lambda_2^3)z, \quad p_3 = -\lambda_2^2 z^4, \quad z = \sqrt{4\lambda_0^2 + \lambda_1^2 + 4\alpha_3 \lambda_0 \lambda_2 - k\lambda_2^2}.$$

Equations of the 2nd family of IMSMs and the equation, which defines the vector field on the elements of this family, differ from (17), (18) in the signs of the coefficients.

It is possible to conclude from (18) that motion on the elements of the family of IMSMs (17) is described by hyperelliptic functions of time. The latter can be reduced to elliptic functions when $\lambda_0 = 0$. Equation (18) in this case assumes the form:

$$\dot{M}_3 = \frac{\sqrt{2\alpha_1 \alpha_3 \sqrt{(k\lambda_2^2 - \lambda_1^2)\lambda_2 \lambda_3} M_3^2 - \alpha_1^2 k \lambda_2 \lambda_3 M_3^4 - \lambda_1^2 + k\lambda_2^2}}{2\alpha_1 \sqrt{-\lambda_2 \lambda_3}},$$

and the equations of the corresponding family of IMSMs differ structurally from equations (17) by the absence of addends, which contain $M_3^{18}, M_3^{16}, M_3^6$ in the 1st, 3rd and 5th equations, respectively.

Equations (16) have a finite number of solutions (zero-dimensional sets), and, as the Maple program “NumberOfSolutions” shows, there shall be 33 such solutions. The sets of nonzero dimension in this case may be obtained if we solve the scrutinized equations with respect, for example, to the parameters λ_1, λ_2 and the phase variables $M_2, \gamma_1, \gamma_2, \gamma_3$. Using the Gröbner bases technique, we have found a series of solutions for equations (16) with respect to the variables indicated. Some of these are given below:

$$\begin{aligned} \gamma_1 &= \pm \frac{\sqrt{\alpha_1 M_1 z}}{2M_3}, \quad \gamma_2 = \frac{\alpha_1 (M_3^2 - M_1^2)}{2M_3} - \alpha_3 M_1, \quad \gamma_3 = \pm \frac{\sqrt{\alpha_1 z}}{2\sqrt{M_1}}, \quad M_2 = 0, \\ \lambda_1 &= \frac{\sqrt{\alpha_1 z} [2\lambda_0 M_1 \pm \lambda_3 (M_1^2 + M_3^2) (\alpha_1 M_3 - \alpha_3 M_1) M_3^2]}{\sqrt{M_1} (\alpha_1 (M_1^2 - M_3^2) + 2\alpha_3 M_1 M_3)}, \\ \lambda_2 &= \frac{[4\lambda_0 M_1 + \alpha_1 \lambda_3 (M_1^2 + M_3^2)^2 M_3] M_3}{\alpha_1 (M_3^2 - M_1^2) - 2\alpha_3 M_1 M_3}. \end{aligned} \tag{19}$$

Here $z = -(2\alpha_3 M_3^3 + \alpha_1 M_1 (M_1^2 + 3M_3^2))$.

Analysis of the solutions has shown that the first four equations define the family of IMSMs of differential equations (12) written in different maps. The vector field on the elements of this family of IMSMs is described by the equations:

$$\dot{M}_1 = 0, \quad \dot{M}_3 = 0. \tag{20}$$

The latter two equations in (19) for λ_1 and λ_2 represent the first integrals of differential equations (20). The integrals here are trivial because the equations themselves are trivial.

3.2 On Bifurcations of Invariant Manifolds

Consider the problem of branching for the obtained families of IMSMs, while restricting the consideration – likewise above – with trivial conditions imposed on the problem parameters.

While following the technique of section 2.2, find the conditions, under which the families of IMSMs, which adjunct to (I9), can be found as the solutions of stationarity equations (I4). To this end, compute the Jacobian of system (I4) on the solution under consideration and, by equating the expression obtained to zero, find the desired conditions. Such conditions, in particular, are (a) $\alpha_1 = 0$, (b) $\lambda_3 = 0$.

In the case, when $\alpha_1 = 0$, we have found several 3-dimensional IMSMs. One of these IMSMs has a rather simple form: $\gamma_1 = \alpha_3 M_2, \gamma_2 = -\alpha_3 M_1, \gamma_3 = 0, \lambda_0 = -\alpha_3 \lambda_2 / 2, \lambda_1 = 0, \alpha_1 = 0$, and, as obvious from the analysis, adjuncts to the 0th solution.

In the case when $\lambda_3 = 0$, we have found one-dimensional and two-dimensional families of IMSMs, which adjunct to the family of IMSMs (I9). One of these families may be obtained when putting $\lambda_3 = 0$ in (I9).

As far as the family of IMSMs (I7) is concerned, we have not managed to conduct analysis for the Jacobian expression computed on the given solution because the obtained expression is rather bulky.

4 On “Resonance” Invariant Manifolds

Consider the problem of obtaining “resonance” invariant manifolds for the systems of differential equations (I1), (I2). To this end, we employ the following procedure.

Let the system of differential equations $\dot{x}_i = X_i(x_1, \dots, x_n)$ assume the two first integrals: $V(x) = c_1, W(x) = c_2$.

Consider some smooth manifold defined by the equations:

$$\varphi_j(x_1, \dots, x_n) = 0 \quad (j = 1, \dots, k). \tag{21}$$

Using the equations $\varphi_j = \varphi_j(x_1, \dots, x_n)$, exclude k variables x_i ($i = 1, \dots, k$) from the first integrals and equations of motion. As a result, the first integrals assume the following form: $\bar{V} = \bar{V}(\varphi_1, \dots, \varphi_k, x_{k+1}, \dots, x_n), \bar{W} = \bar{W}(\varphi_1, \dots, \varphi_k, x_{k+1}, \dots, x_n)$.

Let there exist some polynomial relation, for example,

$$\bar{V}^p(0, \dots, 0, x_{k+1}, \dots, x_n) = \bar{W}^q(0, \dots, 0, x_{k+1}, \dots, x_n), \tag{22}$$

on the manifold (21) between the integrals \bar{V} and \bar{W} , where $p \neq 0, q \neq 0$ are some integer numbers.

Consider the linear bundle of the first integrals $\bar{K} = \bar{V}(\varphi_1, \dots, \varphi_k, x_{k+1}, \dots, x_n) - \mu \bar{W}(\varphi_1, \dots, \varphi_k, x_{k+1}, \dots, x_n)$. The following theorem is valid.

Theorem. If for some values of $\mu = \text{const}$ the system of equations

$$\frac{\partial \bar{K}}{\partial \varphi_j} = \frac{\partial \bar{V}}{\partial \varphi_j} - \mu \frac{\partial \bar{W}}{\partial \varphi_j} = 0 \quad (j = 1, \dots, k),$$

is satisfied for any x_{k+1}, \dots, x_n on the manifold $\varphi_j = 0 \quad (j = 1, \dots, k)$, and relation (22) is valid, then the differential equations have the family of invariant manifolds

$$\varphi_j(x_1, \dots, x_n) = 0 \quad (j = 1, \dots, k), \quad \bar{V}(0, \dots, 0, x_{k+1}, \dots, x_n) = \left(\frac{q}{\mu p}\right)^{\frac{q}{p-q}},$$

on which function \bar{K} assumes a stationary value.

Let us use this theorem for obtaining IMSMs in our case. Consider the following relation between the first integrals of equations (II): $V_0^3 = \lambda V_3^2 \quad (\lambda = \text{const})$.

The latter equality turns into an identity when $\gamma_1 = \gamma_3 = M_1 = M_2 = 0, \gamma_2 = \alpha_1 M_3$ and $\lambda = -\alpha_3^{-1}$. Stationary conditions for the integral $K = V_0 - \mu V_3 \quad (\mu = \text{const})$ with respect to variables $\gamma_1, \gamma_3, M_1, M_2, y = \gamma_2 - \alpha_1 M_3$ hold for $\alpha_1 = 0$.

So, conditions of above theorem are satisfied here, and equations (II) have the family of IMSMs

$$\gamma_1 = \gamma_2 = \gamma_3 = M_1 = M_2 = 0, \quad 3\alpha_3 M_3 = -2\mu^{-1}$$

under the condition, when $\alpha_1 = 0$.

Now consider the relation between the first integrals of equations (I2): $\bar{V}_2^3 = \lambda \bar{V}_3$. The latter equality turns into an identity when $M_1 = M_2 = \gamma_1 = \gamma_3 = 0, \gamma_2 = \alpha_1 M_3/2$ and $\lambda = -27\alpha_1^2/16$. Stationary conditions for the integral $K = \bar{V}_2 - \mu \bar{V}_3$ with respect to the variables $\gamma_1, \gamma_3, M_1, M_2, y = \gamma_2 - \alpha_1 M_3/2$ hold for $\alpha_3 = 0$.

Hence equations (I2) have the family of of IMSM

$$M_1 = M_2 = \gamma_1 = \gamma_3 = 0, \quad \gamma_2 = \frac{1}{2}\alpha_1 M_3, \quad M_3^4 = -\mu^{-1}$$

under the condition when $\alpha_3 = 0$.

5 On Stability of Invariant Manifolds

Let us represent some results of investigation of stability for a series of above IMSMs. For the purpose of investigation, we used the Routh–Lyapunov method. This method allows one to obtain sufficient stability conditions.

Investigate the stability of one of the families of IMSMs (6), for example: $\gamma_1 = \alpha_3 M_2, \gamma_2 = -\alpha_3 M_1, \gamma_3 = 0, M_3 = \lambda_2/\lambda_3$.

In accordance with the indicated method, we introduce deviations $\zeta_1 = \gamma_1 - \alpha_3 M_2, \zeta_2 = \gamma_2 + \alpha_3 M_1, \zeta_3 = \gamma_3, \zeta_4 = M_3 - \lambda_2/\lambda_3$ and write the 2nd variation of integral K (3) in the neighbourhood of manifold under consideration.

The second variation of integral K in terms of deviations has the form:

$$\delta^2 K = -\frac{3}{2}\lambda_2(\zeta_1^2 + \zeta_2^2 + \zeta_3^3 + \alpha_3^2 \zeta_4^2). \tag{23}$$

Obviously, the quadratic form (23) is signdefinite for all $\lambda_2 \neq 0$ and $\alpha_3 \neq 0$. Consequently, due to Zubov’s theorem [11], the elements of the scrutinized family of IMSMs is stable under the condition indicated.

Next, let us investigate stability of the family of IMSMs (7):

$$\begin{aligned} \gamma_1 &= \frac{\sqrt{M_1 z}}{M_3}, \quad \gamma_2 = -2\left[\frac{\alpha_1 M_1^2}{M_3} - \alpha_1 M_3 + 2\alpha_3 M_1\right], \quad \gamma_3 = \frac{\sqrt{z}}{\sqrt{M_1}}, \quad M_2 = 0, \\ z &= \alpha_1(3\alpha_1 M_1 + 2\alpha_3 M_3)M_3^2 - (2\alpha_1 M_1 + 3\alpha_3 M_3)^2 M_1. \end{aligned} \tag{24}$$

The vector field on the elements of this family is described by the equations:

$$\dot{M}_1 = 0, \quad \dot{M}_3 = 0, \tag{25}$$

which are obtained from equations (1) with the use of expressions (24).

From the geometric viewpoint, elements of the family of IMSMs (24) represent some surfaces in R^6 , at each point of which a solution ($M_1 = M_1^0 = \text{const}$, $M_3 = M_3^0 = \text{const}$) of equations (25) is defined.

Consider the problem of stability investigation of solutions, which belong to the elements of the family of IMSMs (24) and correspond to parameters M_1^0, M_3^0 . Likewise in the previous case, this problem is reduced to the investigation of signdefiniteness of the 2nd variation of integral K (3), which has been obtained in the neighbourhood of the solution under investigation.

The second variation of integral K in the neighbourhood of the solution corresponding to M_1^0, M_3^0 , in terms of deviations

$$\begin{aligned} \xi_1 &= \gamma_1 - \frac{\sqrt{M_1^0 z^0}}{M_3^0}, \quad \xi_2 = \gamma_2 + 2\left[\frac{\alpha_1 M_1^{02}}{M_3^0} - \alpha_1 M_3^0 + 2\alpha_3 M_1^0\right], \quad \xi_3 = \gamma_3 - \frac{\sqrt{z^0}}{\sqrt{M_1^0}}, \\ \xi_4 &= M_2, \quad \xi_5 = M_1 - M_1^0, \quad \xi_6 = M_3 - M_3^0, \\ z^0 &= \alpha_1(3\alpha_1 M_1^0 + 2\alpha_3 M_3^0)M_3^{02} - (2\alpha_1 M_1^0 + 3\alpha_3 M_3^0)^2 M_1^0, \end{aligned}$$

writes:

$$\begin{aligned} \delta^2 K &= a_1 \xi_1^2 + a_2 \xi_1 \xi_4 + a_3 \xi_1 \xi_5 + a_4 \xi_1 \xi_6 + a_5 \xi_2^2 + a_6 \xi_2 \xi_4 + a_7 \xi_2 \xi_5 + a_8 \xi_2 \xi_6 \\ &+ a_9 \xi_3^2 + a_{10} \xi_3 \xi_5 + a_{11} \xi_3 \xi_6 + a_{12} \xi_4^2 + a_{13} \xi_4 \xi_6 + a_{14} \xi_5^2 + a_{15} \xi_5 \xi_6 + a_{16} \xi_6^2. \end{aligned}$$

Now let us investigate signdefiniteness $\delta^2 K$ on the linear manifold:

$$\begin{aligned} \delta V_0 &= b_1 \xi_2 + b_2 \xi_4 + b_3 \xi_5 + b_4 \xi_6 = 0, \\ \delta V_2 &= b_5 \xi_1 + b_6 \xi_3 + b_7 \xi_4 + b_8 \xi_5 + b_9 \xi_6 = 0, \\ \delta V_1 &= b_{10} \xi_1 + b_{11} \xi_2 + b_{12} \xi_3 + b_{13} \xi_4 + b_{14} \xi_6 = 0. \end{aligned} \tag{26}$$

Here a_i, b_j are some expressions of $\lambda_0, \lambda_1, \lambda_2, \lambda_3, \alpha_1, \alpha_3, M_1^0, M_3^0$.

Having excluded the variables ξ_1, ξ_2 from $\delta^2 K$ with the use of equations (26) (among which there are only two independent ones), we obtain a quadratic form

of the variables $\xi_3, \xi_4, \xi_5, \xi_6$. The conditions of its signdefiniteness are sufficient conditions of stability of solutions under consideration. When written in the form of the Sylvester inequalities with the use of the denotations $a_1 = M_1^{0^2} - M_3^{0^2}$, $a_2 = 3M_1^{0^2} - 2M_3^{0^2}$, $a_3 = 11M_1^{0^2} - 3M_3^{0^2}$, $a_4 = 4M_1^{0^2} - 3M_3^{0^2}$, $a_5 = 3M_1^{0^2} - M_3^{0^2}$, $a_6 = 27M_1^{0^2} - 5M_3^{0^2}$, these write:

$$\begin{aligned} \Delta_1 &= -\frac{(M_1^{0^2} + M_3^{0^2})(2\lambda_3 M_3^0 + \lambda_2)}{2M_1^{0^2}} > 0, \quad \Delta_2 = \frac{2\lambda_3 M_3^0 + \lambda_2}{M_1^{0^3}} [9\alpha_3^3(2\lambda_3 M_3^0 \\ &+ \lambda_2)M_1^0 M_3^{0^2} + \alpha_1^2(a_3 \lambda_3 M_3^0 + a_4 \lambda_2)M_1^0 + \alpha_1 \alpha_3(a_6 \lambda_3 M_3^0 + 4a_5 \lambda_2)M_3^0] > 0, \\ \Delta_3 &= -\frac{2\alpha_1 \Delta_2}{M_1^{0^2}(2\lambda_3 M_3^0 + \lambda_2)} [\alpha_3(2\lambda_3 M_3^0 + \lambda_2)(\lambda_3 M_3^0 + 2\lambda_2)M_1^0 M_3^0 + \alpha_1(a_1(\lambda_2^2 \\ &+ \lambda_3^2 M_3^{0^2}) + a_2 \lambda_2 \lambda_3 M_3^0)] > 0, \quad \Delta_4 = -\left[\frac{(2\lambda_3 M_3^0 + \lambda_2) \Delta_3}{M_1^{0^2} M_3^0 \Delta_2} \right]^2 \left[(\alpha_1^3(5M_1^{0^2} - 9M_3^{0^2}) \right. \\ &\times M_1^0 + 18\alpha_3^3 M_3^{0^3})M_1^{0^3} + \alpha_1 \alpha_3(2\alpha_1(12M_1^{0^2} - 11M_3^{0^2})M_1^{0^3} M_3^0 - \alpha_3((M_3^{0^2} \\ &+ 15M_1^{0^2})M_3^{0^2} - 36M_1^{0^4})) \Big] > 0. \end{aligned} \tag{27}$$

To solve the system of inequalities (27) we have used the *Mathematica* program “Reduce”. We cannot represent the program result in its complete form because it is rather bulky. Below you can see only some of the obtained compatibility conditions of inequalities (27):

$$\begin{aligned} &\alpha_1 \neq 0 \wedge M_1^0 \neq 0 \\ &\wedge \left[\left(\alpha_3 = 0 \wedge \left(\frac{a_1 \lambda_3 M_3^0}{a_2} + \lambda_2 < 0 \wedge ((\lambda_3 > 0 \wedge M_1^0 + M_3^0 = 0) \right. \right. \right. \\ &\quad \left. \left. \left. \vee (\lambda_3 < 0 \wedge M_1^0 = M_3^0) \right) \right) \right) \\ &\vee \left(\alpha_3 \neq 0 \wedge \left(\frac{\alpha_3(2\lambda_3 M_3^0 + 5\lambda_2)M_1^0 M_3^0 + \alpha_1(a_1 \lambda_3 M_3^0 + a_2 \lambda_2)}{a_1 \alpha_1 + 5\alpha_3 M_1^0 M_3^0} < 0 \right. \right. \\ &\quad \wedge \left((M_3^0 = \frac{(\alpha_3 + \sqrt{\alpha_1^2 + \alpha_3^2})M_1^0}{\alpha_1} \wedge \lambda_3 < 0) \right. \\ &\quad \left. \left. \left. \vee (M_3^0 = \frac{(\alpha_3 - \sqrt{\alpha_1^2 + \alpha_3^2})M_1^0}{\alpha_1} = 0 \wedge \lambda_3 > 0) \right) \right) \right) \Big]. \end{aligned}$$

So, stable in the sense of Lyapunov shall be only those considered solutions, which belong to the elements of the family of IMSMs (24), for which the parameters M_1^0, M_3^0 satisfy the obtained conditions.

A similar stability investigation has also been conducted for the family of IMSMs (19). Here, sufficient stability conditions have been obtained for the solutions, which belong to the elements of the given family of IMSMs.

References

1. Borisov, A.V., Mamayev, I.S., Sokolov, V.V.: A new integrable case on $\mathfrak{so}(4)$. Dokl. Phys. 46(12), 888–889 (2001)
2. Sokolov, V.V.: A new integrable case for the Kirchhoff equation. Theoret. and Math. Phys. 129(1), 1335–1340 (2001)
3. Sokolov, V.V.: On a class of quadratic Hamiltonians on $\mathfrak{so}(4)$. Dokl. Math. 69, 108–111 (2004)
4. Bogoyavlensky, O.I.: Breaking Solitons. Nonlinear Integrable Equations. Moscow, Nauka (1991)
5. Smirnov, A.V.: Systems of $sl(2, \mathbb{C})$ tops as two-particle systems. Theoret. and Math. Phys. 157(1), 1370–1382 (2008)
6. Tsiganov, A.V., Goremykin, O.V.: Integrable systems on $\mathfrak{so}(4)$ related with XXX spin chains with boundaries. J. Phys. A: Math. Gen. 37, 4843–4849 (2004)
7. Sokolov, V.V., Wolf, T.: Integrable quadratic classical Hamiltonians on $\mathfrak{so}(4)$ and $\mathfrak{so}(3,1)$. J. Phys. A: Mat. Gen. 39, 1915–1926 (2006)
8. Rumyantsev, V.V.: A comparison of three methods of constructing Lyapunov functions. J. Appl. Math. Mech. 59(6), 873–877 (1995)
9. Irtegov, V.D.: Invariant Manifolds of Steady-State Motions and Their Stability, Nauka, Novosibirsk (1985)
10. Karapetyan, A.V.: The Stability of Steady Motions, Moscow, URSS (1998)
11. Zubov, V.I.: Stability of integrable manifolds. Differential equations 13(9), 1720–1722 (1977)

On the Complexity of Reliable Root Approximation

Michael Kerber

Max-Planck Institut für Informatik, Saarbrücken, Germany
mkerber@mpi-inf.mpg.de

Abstract. This work addresses the problem of computing a certified ϵ -approximation of all real roots of a square-free integer polynomial. We prove an upper bound for its bit complexity, by analyzing an algorithm that first computes isolating intervals for the roots, and subsequently refines them using Abbott’s Quadratic Interval Refinement method. We exploit the eventual quadratic convergence of the method. The threshold for an interval width with guaranteed quadratic convergence speed is bounded by relating it to well-known algebraic quantities.

1 Introduction

Computing the roots of a univariate polynomial is one of the most prominent problems in Computer Algebra. For the case that only real roots are of interest, several subdivision approaches, based on Descartes’ rule of sign or on Sturm’s Theorem have been introduced [6, 14]. Their output consists of a set of disjoint intervals, each containing exactly one root of the polynomial, and vice versa, each root is contained in one of the intervals; they are also called *isolating intervals*. These subdivision solvers constitute a popular method for root finding, primarily as they return a certified output (no root is lost, no interval contains several roots). Also, they are relatively easy to implement, and have shown good practical performance. Real root solving is a cornerstone, for instance, for the computation of Cylindrical Algebraic Decomposition [4], for related problems such as topology computation [11, 8] and arrangement computation [10], and many more.

In this work, we will investigate the cost of computing isolating intervals, and subsequently refining them until their width falls below ϵ . An equivalent description is to approximate all roots to a precision of ϵ . It should not be surprising that this problem frequently appears in concrete applications – for instance, when comparing the roots of two polynomials, or when evaluating the sign of an algebraic expression that depends on a root of a polynomial.

While the (worst-case) complexity of the root isolation process has been studied extensively for various isolation methods [9, 12, 16], similar results seem not to be available yet for the subsequent refinement process. Our work will provide a complexity analysis with the following main result. Let $f := \sum_{i=0}^p a_i x^i \in \mathbb{Z}[x]$ be a polynomial of degree p , with simple roots and $|a_i| < 2^\sigma$ for each coefficient a_i . For $\epsilon > 0$, computing isolating intervals of width at most ϵ for all roots requires in the worst-case

$$\tilde{O}(p^4 \sigma^2 + p^3 \log \epsilon^{-1}), \tag{1}$$

bit operations, where \tilde{O} means that logarithmic factors in p and σ are neglected.

We achieve our bound by analyzing the *Quadratic Interval Refinement* (*qir*) method to refine isolating intervals, introduced by Abbott [1]. This method can be considered as a hybrid of bisection and (an interval version of) the secant method. We will discuss the algorithm in detail in Section 3. As Abbott has already pointed out, the method initially behaves like naive bisection (linear convergence), but once the interval falls below a certain width, the number of newly obtained bits is doubled in every step (which basically means quadratic convergence). In our analysis, we split the sequence of *qir* steps into an initial sequence where we assume bisections, and a quadratic sequence where the root is rapidly approximated. We will show that the sum of the cost of all initial sequences is bounded by the first summand of (1) (which also bounds the cost of the root isolation), and that the second summand is caused by the cost of the quadratic sequence. It is remarkable that our analysis profits from considering all (real) roots of f ; when restricting to a single root of f , we are able to decrease only the second summand by a factor of p , even if the root is already given by an isolating interval.

The reader might wonder at this point why not using a more prominent algorithm like the famous *Newton iteration* instead of the *qir* method. A problem in Newton's method lies in the choice of a starting value – an unfortunate one leads to a diverging sequence. A solution is to perform bisections initially to produce an interval where convergence of Newton's method is guaranteed, and then to switch to Newton iteration manually. However, this manual switch depends on theoretical worst-case bounds for valid starting values of Newton's method, thus more bisections than actually necessary are performed in the average case. The *qir* method, in contrast, switches adaptively as soon as possible, independently of the worst-case bounds that are introduced only for the analysis.

Dekker [7] presented a method which, similarly to the *qir*, combines bisections and the secant method. Brent [3] combines Dekker's method with inverse quadratic interpolation. Superlinear convergence can also be guaranteed for this method. However, a problem in Dekker's approach is the growth in the bitsize of the iteration values – it appears unclear to the author how to choose a suitable working precision in each substep to avoid a too big coefficient swell-up while still guaranteeing fast convergence. The same holds true for Brent's method, and additionally, an analysis seems to be even more involved as it even adds more ingredients to Dekker's method. The *qir* method guarantees a minimal growth in the bitsizes, since all intervals are of the form $[\frac{a}{2^\ell}, \frac{a+1}{2^\ell}]$ (with $a, \ell \in \mathbb{Z}$), thus the bitsize of the boundaries is proportional to the interval width, what is the best one can hope for.

The simpleness of the *qir* method also make this approach attractive for concrete implementation. It is used both in the COCOA library [1] and the (experimental) algebraic kernel of the CGAL library [8] (used, for instance, in [11, 10]). Its application is also attested in [8]. In this work, however, we focus on the complexity analysis, and do not address its practical performance.

This paper is structured as follows: In Section 2, we give a rough overview about real root isolation algorithms, and their complexity. Section 3 revises the *qir* method. Our complexity bound (1) is proved in Section 4. We conclude in Section 5.

¹ <http://http://cocoa.dima.unige.it/>

² <http://www.cgal.org>

Notation

It will be convenient to fix some notation. Throughout this article, let $f = \sum_{i=0}^p a_i x^i$ be a square-free polynomial (i.e., without multiple roots) of degree p , with integer coefficients a_i of bitsize σ , that means, $|a_i| \leq 2^\sigma$. The complex roots of f are denoted by $\alpha_1, \dots, \alpha_p$, and we assume exactly the first s roots $\alpha_1, \dots, \alpha_s$ to be real.

Also, let $0 < \epsilon < 1$ be fixed, and set $L := \log \frac{1}{\epsilon}$. We write $M(n)$ for the cost of multiplying two integers of bitsize n , and assume that $M(n) = O(n \log n \log \log n)$, according to the fast multiplication algorithm by Schönhage and Strassen [15]. To keep the complexity bound handleable, we will often neglect logarithmic factors in p and σ and denote such complexity bounds by $\tilde{O}(\cdot)$. As an example, $M(n) = \tilde{O}(n)$. Finally, for $I = (c, d)$, we denote by $w(I) := d - c$ its width.

2 Root Isolation

Several approaches have been investigated for the root isolation problem. They all accept the square-free polynomial f as input, and produce a list of s isolating intervals for $\alpha_1, \dots, \alpha_s$. A considerable body of literature has appeared about this problem (a small subset is [5,6,14,16]); it is not the scope of this work to discuss them in detail – still, their worst-case bound is of importance.

Theorem 1. *Computing isolating intervals for the real roots of f requires at most $\tilde{O}(n^4 \sigma^2)$ bit operations in the worst-case (using fast arithmetic). Moreover, each isolating interval is of the form $(\frac{a}{2^\ell}, \frac{a+1}{2^\ell})$ with $a, \ell \in \mathbb{Z}$ and $\log |\frac{a}{2^\ell}| = O(\sigma)$.*

The complexity bounds have been proved for root isolation based on *Sturm sequences* [9], and based on *Descartes' rule of signs* [12]. The special form of the isolating intervals is a consequence of the subdivision that is initially started with an interval $[-2^{O(\sigma)}, 2^{O(\sigma)}]$ that covers all real roots of f (compare [2, §10.1]).

We remark that the *Continued fraction algorithm* (introduced in [5]) usually perform best in practice among the available modern root solvers, although the best known bound in the literature seems to be $\tilde{O}(n^5 \tau^2)$ [16]. See [13] for a recent experimental comparison on various modern root solvers.

3 Abbott's Quadratic Interval Refinement

Everybody knows about the most naive method for refining isolating intervals – the *bisection method*. Given an isolating interval (c, d) , evaluate f at the midpoint $m = \frac{c+d}{2}$. If $f(m) = 0$, the root is found exactly. Otherwise, either (c, m) or (m, d) is chosen as refined isolating interval, depending on where the sign change takes place. Clearly, the isolating interval is halved in every step which means that one bit of precision is added per bisection.

The analysis of the complexity for the bisection is also straight-forward. The crucial operation is to evaluate f at m , the number of arithmetic operations is linear.

Algorithm 1. Quadratic interval refinement

```

1: procedure QIR( $f, I = (c, d), N$ )  $\triangleright$  Returns a pair  $(J, N_{new})$ , with  $J$  the refined interval
2:   if  $N = 2$ , return (BISECTION( $f, I, 4$ )).
3:    $w \leftarrow \frac{d-c}{N}$ 
4:    $m' \leftarrow c + \text{round}(N \frac{f(c)}{f(c)-f(d)})w$   $\triangleright m = c + \frac{f(c)}{f(c)-f(d)}(d-c)$ 
5:    $s \leftarrow \text{sgn}(f(m'))$ 
6:   if  $s = 0$ , return ( $[m', m'], \infty$ )
7:   if  $s = \text{sgn}(f(c))$  and  $\text{sgn}(f(m' + w)) = \text{sgn}(f(d))$ , return ( $(m', m' + w), N^2$ )
8:   if  $s = \text{sgn}(f(d))$  and  $\text{sgn}(f(m' - w)) = \text{sgn}(f(c))$ , return ( $(m' - w, m'), N^2$ )
9:   Otherwise, return ( $I, \sqrt{N}$ ).
10: end procedure

```

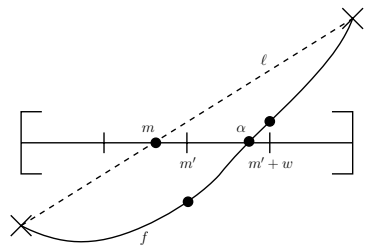
If τ denotes the bitsize of c and d , then the bisection has to deal with bitsizes up to $O(\sigma + p\tau)$ during the evaluation, and thus the number of bit operations is bounded by

$$O(pM(\sigma + p\tau)).$$

What if we did bisection until the interval gets smaller than ϵ ? We would have to perform up to $\sigma + L$ bisection steps (the initial σ bisections to make its width smaller than one), and the interval boundaries would grow to bitsize $\sigma + L$. Thus, one would arrive at a total complexity of $O(p(\sigma + L)M(p(\sigma + L))) = \tilde{O}(p^2(\sigma + L)^2)$, with an additional factor of p when doing this for each root. Not surprisingly, this is inferior to **(I)** since L appears quadratically.

A more efficient way of refining the isolating interval has been presented with Abbott's *Quadratic Interval Refinement method* **(II)**; we call it *qir* from now. Consider an isolating interval $I = (c, d)$ for a root α , and let ℓ be the secant through the points $(c, f(c))$ and $(d, f(d)) \in \mathbb{R}^2$. If I is small enough, f should almost look like the line ℓ over I , and thus, the intersection point m of ℓ with the x -axis should be close to α .

This idea leads to the following algorithm: Having an additional integer N as input, subdivide I (conceptually) by $N + 1$ equidistant grid points (with distance $w := \frac{w(I)}{N}$). Then, compute m' , the closest grid point to m , and evaluate $f(m')$. Depending on its sign, evaluate the sign of either the left or right neighboring grid point. If the sign changes from m' to $m' \pm w$, choose it as new isolating interval (this refines by a factor of N) and set N to N^2 for the next *qir* call. Otherwise, keep I as isolating interval and set N to \sqrt{N} for the next call. If $N = 2$, perform one bisection step. See also Algorithm **(I)** for a pseudo-code description.



Successful *qir* instance for $N = 4$

We assume that N is initially set to 4 for an isolating interval returned by a root isolation algorithm, and that the method QIR is always called with the parameter N that has been returned in the previous call for the given interval.

Different from Abbott's original formulation, a call of QIR does not necessarily refine the isolating interval. However, in this case, N is decreased as a side effect, and at the latest when $N = 2$, the method will refine the interval eventually.

Algorithm 2. Root isolation with refinement

```

1: procedure ISOLATE_AND_REFINE( $f, \epsilon$ )
2:    $I_1, \dots, I_s \leftarrow \text{ISOLATE}(f)$  ▷ see Section 2
3:   for  $k \in \{1, \dots, s\}$  do
4:      $N \leftarrow 4$ 
5:     while  $\text{width}(I_i) > \epsilon$  do  $(I_i, N) \leftarrow \text{QIR}(f, I_i, N)$ 
6:   end for
7:   return  $I_1, \dots, I_s$ 
8: end procedure

```

Definition 1. A qir call $(J, N_2) \leftarrow \text{QIR}(f, I, N_1)$ succeeds, if $J \subsetneq I$, and it fails, if $J = I$. Equivalently, the qir call succeeds, if and only if $N_2 > N_1$.

For one qir call (successful or not), one has to perform only $O(p)$ arithmetic operations to evaluate f at m' and $m' \pm w$, and perform another constant number of arithmetic operations. The bitsize of m' and $m' \pm w$ is bounded by $O(\log N + \tau)$ where τ is the maximal bitsize of c and d .

It is easy to see that $\log N \in O(\tau)$, assuming that the qir is initially started with $N = 4$: if a qir call with $N > 4$ subintervals is started, there must have been a successful qir call for \sqrt{N} . Thus, the width of the interval is at most $\frac{1}{\sqrt{N}}$, and the bitsize of either c or d must be at least $\log \sqrt{N} = \frac{1}{2} \log N$.

After all, the cost of one qir call is thus bounded by

$$O(pM(\sigma + p\tau)),$$

which is equal to the cost of one bisection step. We remark that one successful qir step yields exactly the same result as $\log N$ bisections, so that the isolating interval remains of the form $(\frac{a}{2^\ell}, \frac{a+1}{2^\ell})$ if the initial interval was of this type.

4 Analysis of Root Refinement

We prove the bound given in (1). For that, we analyze the complexity of this straightforward algorithm: Apply QIR to each isolating interval until its width falls below ϵ (Algorithm 2).

Definition 2. Let α be a root of f for which Step 2 of Algorithm 2 returned the isolating interval I_0 . The qir sequence (s_0, \dots, s_n) for α , is defined as

$$s_0 := (I_0, 4) \quad s_i := (I_i, N_i) := \text{QIR}(f, I_{i-1}, N_{i-1}) \quad \text{for } i \geq 1$$

where I_n is the first index such that $w(I_n) \leq \epsilon$. We say that $s_{i-1} \xrightarrow{\text{QIR}} s_i$ succeeds if $\text{QIR}(f, I_{i-1}, N_{i-1})$ succeeds, and that $s_{i-1} \xrightarrow{\text{QIR}} s_i$ fails otherwise.

The qir sequence for α is split into two subsequences, according to the value M_α defined in the next lemma. M_α will turn out to be an upper bound for the width of the isolating interval of α that ensures quadratic convergence. We will prove this in Section 4.2, but we already show two simple properties of M_α .

Lemma 1. *Let $\alpha \in \mathbb{C}$ be a root α of f . We define*

$$M_\alpha := \frac{|f'(\alpha)|}{2ep^3 2^\sigma \max\{|\alpha|, 1\}^{p-1}}$$

with $e \approx 2.718$. It holds that

1. $0 < M_\alpha < \frac{1}{p}$
2. Let $\mu \in \mathbb{C}$ be such that $|\alpha - \mu| < M_\alpha$. Then

$$M_\alpha < \frac{|f'(\alpha)|}{2|f''(\mu)|}.$$

Proof. We bound $|f'(\alpha)|$ from above by the following

$$|f'(\alpha)| = \left| \sum_{i=1}^p i a_i \alpha^{i-1} \right| \leq p 2^\sigma \sum_{i=0}^{p-1} \max\{|\alpha|, 1\}^i \leq p 2^{2\sigma} p \max\{|\alpha|, 1\}^{p-1}$$

which proves the first claim. For the second, we bound $|f''(\mu)|$ from above:

$$\begin{aligned} |f''(\mu)| &= \left| \sum_{i=2}^p i(i-1) a_i \mu^{i-2} \right| \leq p^2 2^\sigma \sum_{i=0}^{p-2} \max\{|\mu|, 1\}^i \\ &\leq p^2 2^\sigma \sum_{i=0}^{p-2} ((1 + M_\alpha) \max\{|\alpha|, 1\})^i \\ &\leq p^3 2^\sigma (1 + M_\alpha)^{p-2} \max\{|\alpha|, 1\}^{p-2} < p^3 2^\sigma \underbrace{\left(1 + \frac{1}{p}\right)^p}_{< e} \max\{|\alpha|, 1\}^{p-1} \end{aligned}$$

This shows that

$$\frac{|f'(\alpha)|}{2|f''(\mu)|} > \frac{|f'(\alpha)|}{2e \cdot p^3 2^\sigma \max\{|\alpha|, 1\}^{p-1}} = M_\alpha \quad \square$$

Definition 3. *Let (s_0, \dots, s_n) be the qir sequence for α . Let k be the minimal index such that $s_k = (I_k, N_k) \xrightarrow{\text{QIR}} s_{k+1}$ succeeds, and $w(I_k) \leq M_\alpha$. We call the sequence (s_0, \dots, s_k) the initial sequence, and (s_k, \dots, s_n) the quadratic sequence.*

In other words, the quadratic sequence is the maximal qir sequence that only contains intervals of size at most M_α , and starts with a successful qir step. In the next two subsections, we will bound the cost of the initial sequence and the quadratic sequence separately.

4.1 Cost of the Initial Sequence

Lemma 2. *Let I be an isolating interval for α . The cost of the initial sequence of α is bounded by*

$$\tilde{O}(p^2(\sigma + \log \frac{1}{M_\alpha})^2)$$

Proof. Let n_q be the number of qir calls until I is refined such that $w(I) < M_\alpha$. Likewise, let n_b be the number of bisections that would be needed to refine I to size M_α . Note that $n_b = O(\sigma + \log \frac{1}{M_\alpha})$.

A successful qir call for some $N = 2^{2^i}$ yields the same accuracy as 2^i bisections, and can only cause up to $i + 1$ subsequent failing qir calls before the next successful qir call. With that argument, it follows that $n_q \leq 2n_b$, so the number of qir calls is in $O(\sigma + \log \frac{1}{M_\alpha})$.

To bound the bitsizes, let N_e be the value of N in the last successful qir call of the initial sequence. It holds that $\log N_e \leq 2n_b$, since otherwise, the preceding qir call would have yielded as much accuracy as $\log \sqrt{N_e} > n_b$ bisections, and the initial sequence would have stopped earlier. Hence, the width of the final interval is at least $\frac{M_\alpha}{N_e}$, and the interval boundaries have bit complexity

$$\log \frac{N_e}{M_\alpha} \leq 2n_b + \log \frac{1}{M_\alpha} = O(\sigma + \log \frac{1}{M_\alpha})$$

Therefore, the bitsizes of the qir calls are bounded by $O(p(\sigma + \log \frac{1}{M_\alpha}))$, which proves the claim. \square

It remains to bound the quantity $\log \frac{1}{M_\alpha}$. We do this simultaneously for all real roots of the polynomial, according to the following theorem.

Theorem 2. *Let $\alpha_1, \dots, \alpha_s$ be the real roots of f . Then,*

$$\sum_{i=1}^s \log \frac{1}{M_{\alpha_i}} = O(p(\sigma + \log p)).$$

Proof. Recall that $0 < M_\alpha < 1$ for each (complex) root α , so $\log \frac{1}{M_\alpha} > 0$, and we can bound:

$$\begin{aligned} \sum_{i=1}^s \log \frac{1}{M_{\alpha_i}} &\leq \sum_{i=1}^p \log \frac{1}{M_{\alpha_i}} = \log \frac{\prod_{i=1}^p 2e \cdot p^3 2^\sigma \max\{|\alpha_i|, 1\}^{p-1}}{|\prod_{i=1}^p f'(\alpha_i)|} \\ &= p \log(2e) + 3p \log p + p\sigma + (p-1) \log \prod_{i=1}^p \max\{|\alpha_i|, 1\} - \log \left| \prod_{i=1}^p f'(\alpha_i) \right| \end{aligned}$$

For both occurring products, we can apply well-known bounds from Algebra. For the first one, note that

$$\text{Mea}(f) := |a_p| \prod_{i=1}^p \max\{|\alpha_i|, 1\}$$

is the *Mahler measure* of f , and it holds that ([17] Lemma 4.14, [2] Prop.10.9)

$$\text{Mea}(f) \leq \|f\|_2 \leq \sqrt{p+1} \cdot \|f\|_\infty \leq \sqrt{p+1} \cdot 2^\sigma.$$

So, $\log \prod_{i=1}^p \max\{|\alpha_i|, 1\} = \log \left(\frac{1}{|a_p|} \text{Mea}(f) \right) \leq \log \text{Mea}(f) = O(\log p + \sigma)$

The second product is related to the resultant of f and f' by the following identity [2] Thm.4.16], [17] Thm.6.15]

$$\text{res}(f, f') = a_p^{p-1} \prod_{i=1}^p f'(\alpha_i).$$

In particular, the right hand side yields an integer. It follows

$$-\log \left| \prod_{i=1}^p f'(\alpha_i) \right| = \log |a_p^{p-1}| - \log \underbrace{|\text{res}(f, f')|}_{\geq 1} < (p-1) \log |a_p| = O(p\sigma).$$

Finally, we can estimate

$$\begin{aligned} \sum_{i=1}^s \log \frac{1}{M_{\alpha_i}} &\leq O(p(\sigma + \log p)) + (p-1) \underbrace{\log \prod_{i=1}^p \max\{|\alpha_i|, 1\}}_{=O(\log p + \sigma)} - \underbrace{\log \left| \prod_{i=1}^p f'(\alpha_i) \right|}_{=O(p\sigma)} \\ &= O(p(\sigma + \log p)). \quad \square \end{aligned}$$

Corollary 1. *The total computation cost for all initial sequences is $\tilde{O}(p^4\sigma^2)$.*

Proof. Combining Lemma 2 and Theorem 2 we get total costs of

$$\tilde{O}\left(\sum_{i=1}^s p^2(\sigma + \log \frac{1}{M_{\alpha_i}})^2\right) = \tilde{O}(p^3\sigma^2 + p^2\left(\sum_{i=1}^s \log \frac{1}{M_{\alpha_i}}\right)^2) = \tilde{O}(p^4\sigma^2) \quad \square$$

Corollary 1 shows that refining all isolating intervals to width M_α does not increase the complexity bound to the initial root isolation.

4.2 Cost of the Quadratic Sequence

In the initial sequence, we have assumed that the qir sequence behaves roughly as the bisection method. As soon as the isolating interval becomes smaller than M_α , we can prove that N is squared in (almost) every step, which leads to quadratic convergence of the interval width. We start with a simple criterion that guarantees a successful qir call.

Lemma 3. *Let $I = (c, d)$ be an isolating interval of α , with $w(I) = \delta$, and consider the qir call $\text{QIR}(f, I, N)$ for some N . Let $m := c + \frac{f(c)}{f(c)-f(d)}(d-c)$ be defined as in the qir method (Algorithm 1). If $|m - \alpha| < \frac{\delta}{2N}$, the qir call succeeds.*

Proof. Recall that I is conceptually subdivided into N subintervals of same width, and that m' is chosen as the grid point closest to m . Let J be the subinterval that contains α , and J' be the subinterval that contains m . If $J = J'$, then one of the endpoints of J' is chosen as m' , so the qir call succeeds. If $J \neq J'$, they must be adjacent, since otherwise, $|m - \alpha| > \frac{\delta}{N}$. W.l.o.g., assume that $m < \alpha$, then m must be in the right half of J , because otherwise $|m - \alpha| > \frac{\delta}{2N}$. Thus, m' is chosen as the right endpoint of J' which is the left endpoint of J . Therefore, the qir call succeeds. \square

We need to investigate the distance between the interpolation point m and the root α . The next theorem shows that this distance depends quadratically on the width of the isolating interval, once it is smaller than M_α . This is basically analogous to Newton's iteration, for which a similar theorem is shown.

Theorem 3. *Let (c, d) be an isolating interval for α of width $\delta < M_\alpha$. Then $|m - \alpha| < \frac{\delta^2}{2M_\alpha}$.*

Proof. We consider the Taylor expansion of f at α . For a given $x \in [c, d]$, we have

$$f(x) = f'(\alpha)(x - \alpha) + \frac{1}{2}f''(\tilde{\alpha})(x - \alpha)^2$$

with some $\tilde{\alpha} \in [x, \alpha]$ or $[\alpha, x]$. Thus, we can simplify

$$\begin{aligned} |m - \alpha| &= \left| \frac{f(d)(c - \alpha) - f(c)(d - \alpha)}{f(d) - f(c)} \right| \\ &= \left| \frac{\frac{1}{2}(f''(\tilde{\alpha}_1)(d - \alpha)^2(c - \alpha) - f''(\tilde{\alpha}_2)(c - \alpha)^2(d - \alpha))}{f(d) - f(c)} \right| \\ &\leq \frac{1}{2}|d - \alpha||c - \alpha| \cdot \frac{|f''(\tilde{\alpha}_1)|(d - \alpha) + |f''(\tilde{\alpha}_2)|(\alpha - c)}{|f(d) - f(c)|} \\ &\leq \frac{1}{2}\delta^2 \max\{|f''(\tilde{\alpha}_1)|, |f''(\tilde{\alpha}_2)|\} \frac{(d - \alpha) + (\alpha - c)}{|f(d) - f(c)|} \\ &= \frac{\delta^2 \max\{|f''(\tilde{\alpha}_1)|, |f''(\tilde{\alpha}_2)|\}}{2|f'(\nu)|} \end{aligned}$$

for some $\nu \in (c, d)$. The Taylor expansion of f' yields $f'(\nu) = f'(\alpha) + f''(\tilde{\nu})(\nu - \alpha)$ with $\tilde{\nu} \in (c, d)$. Since $\delta \leq M_\alpha$, it follows with Lemma \square

$$|f''(\tilde{\nu})(\nu - \alpha)| \leq |f''(\tilde{\nu})|M_\alpha \leq \frac{1}{2}|f'(\alpha)|.$$

Therefore $|f'(\nu)| > \frac{1}{2}|f'(\alpha)|$, and it follows again with Lemma \square that

$$|m - \alpha| \leq \frac{\delta^2 \max\{|f''(\tilde{\alpha}_1)|, |f''(\tilde{\alpha}_2)|\}}{|f'(\alpha)|} = \frac{\delta^2}{2 \frac{|f'(\alpha)|}{\max\{|f''(\tilde{\alpha}_1)|, |f''(\tilde{\alpha}_2)|\}}} < \frac{\delta^2}{2M_\alpha}. \quad \square$$

We apply this theorem on the quadratic sequence.

Corollary 2. *Let I_j be an isolating interval for α of width $\delta_j \leq \frac{1}{N_j}M_\alpha$. Then, each call of the qir sequence $(I_j, N_j) \xrightarrow{\text{QIR}} (I_{j+1}, N_{j+1}) \xrightarrow{\text{QIR}} \dots$ succeeds.*

Proof. We do induction on i . Assume (for $i \geq 0$) that the first i calls succeeded. Then, it is easily shown that $\delta_{j+i} := w(I_{j+i}) = \frac{N_j \delta_j}{N_{j+i}} < \frac{M_\alpha}{N_{j+i}}$ (by another induction, and exploiting that $N_{j+i}^2 = N_{j+i+1}$). Using Theorem 3 we have that

$$|m - \alpha| \leq \delta_{j+i}^2 \frac{1}{2M_\alpha} \leq \delta_{j+i} \frac{M_\alpha}{N_{j+i}} \frac{1}{2M_\alpha} = \frac{1}{2} \frac{\delta_{j+i}}{N_{j+i}}.$$

By Proposition 3 this is enough to guarantee success for the qir method. □

Corollary 3. *In the quadratic sequence, there is at most one failing qir call.*

Proof. Let $(I_i, N_i) \xrightarrow{\text{QIR}} (I_{i+1}, N_{i+1})$ be the first failing qir call in the quadratic sequence. Since the quadratic sequence starts with a successful qir call, the predecessor $(I_{i-1}, N_{i-1}) \xrightarrow{\text{QIR}} (I_i, N_i)$ is also part of quadratic sequence, and succeeds. Thus we have the sequence

$$(I_{i-1}, N_{i-1}) \xrightarrow[\text{QIR}]{\text{Success}} (I_i, N_i) \xrightarrow[\text{QIR}]{\text{Fail}} (I_{i+1}, N_{i+1}) \xrightarrow{\text{QIR}} \dots$$

One observes that $w(I_{i+1}) = w(I_i) = \frac{w(I_{i-1})}{N_{i-1}} \leq \frac{M_\alpha}{N_{i-1}}$, and $N_{i+1} = \sqrt{N_i} = \sqrt{N_{i-1}^2} = N_{i-1}$. By Corollary 2 all further qir calls succeed. □

If the quadratic sequence starts with a bisection (i.e., $N = 2$ initially), no failing qir call occurs. Otherwise, the single failing step is due to the fact that the quadratic sequence might start with a too big value of N , just because the algorithm was “too lucky” during the initial sequence.

Let $(I_{i-1}, N_{i-1}) \xrightarrow{\text{QIR}} (I_i, N_i)$ be the failing qir call in the quadratic sequence. Since for any $k \geq 0$, $w(I_{i+k}) = \frac{N_i w(I_i)}{N_{i+k}}$, it follows that

$$w(I_{i+k+1}) = \frac{w(I_{i+k})^2}{N_k \cdot w(I_k)}.$$

That means, the interval width decreases quadratically in each step (up to the constant $N_k \cdot w(I_k)$) which ultimately justifies the term “quadratic” in the Quadratic Interval Refinement method (the idea of our exposition was already sketched in Abbott’s original work [11]).

Lemma 4. *The number of bit operations in the quadratic sequence of a root α is bounded by*

$$\tilde{O}(p^2 \log L(\sigma + \log \frac{1}{M_\alpha}) + p^2 L).$$

Proof. By Corollary 3 the quadratic sequence consists of at most $\log L + 1$ qir calls, since N is doubled in each step, except the possible failing step. The bitsize in the first qir call of the sequence is $O(p(\sigma + \log \frac{1}{M_\alpha}))$, and increases by at most 2^i after the i -th iteration. Therefore, the complexity of the quadratic sequence is given by

$$\begin{aligned} O\left(\sum_{i=1}^{\log L+1} p \cdot M(p(\sigma + \log \frac{1}{M_\alpha} + 2^i))\right) &= \tilde{O}\left(p^2 \sum_{i=1}^{\log L+1} \sigma + \log \frac{1}{M_\alpha} + 2^i\right) \\ &= \tilde{O}\left(p^2 \log L(\sigma + \log \frac{1}{M_\alpha}) + p^2 \sum_{i=1}^{\log L+1} 2^i\right) = \tilde{O}(p^2 \log L(\sigma + \log \frac{1}{M_\alpha}) + p^2 L) \end{aligned} \quad \square$$

Corollary 4. *The total cost of all quadratic sequences for the real roots $\alpha_1, \dots, \alpha_s$ of f is bounded by*

$$\tilde{O}(p^3 \sigma \log L + p^3 L).$$

Proof. We combine Lemma 4 and Theorem 2 to obtain

$$\sum_{i=1}^s \tilde{O}(p^2 \log L(\sigma + \log \frac{1}{M_\alpha}) + p^2 L) = \tilde{O}(p^3 \sigma \log L + p^2 \log L \underbrace{\sum_{i=1}^s \log \frac{1}{M_\alpha}}_{=O(p(\sigma + \log p))} + p^3 L) \quad \square$$

Combining the cost for root isolation (Theorem 1) with the cost of the initial sequences (Corollary 1) and the cost of the quadratic sequences (Corollary 4) proves the main result:

Theorem 4. *Isolating the real roots of f , and computing an isolating interval of width at most ϵ for each root using Algorithm 2 requires*

$$\tilde{O}(p^4 \sigma^2 + p^3(L + \sigma \log L)) = \tilde{O}(p^4 \sigma^2 + p^3 L)$$

bit operations.

Proof. We only have to argue why the summand $p^3 \sigma \log L$ can never dominate the other two. If $p^4 \sigma^2$ was dominated by $p^3 \sigma \log L$, $\log L$ would dominate $p\sigma$, and in particular L would dominate 2^σ . If also $p^3 L$ was dominated by $p^3 \sigma \log L$, then $\frac{L}{\log L}$ is dominated by σ , so L is dominated by $\sigma^{1+\gamma}$ for any $\gamma > 0$. Contradiction. \square

If only one isolating interval is refined, our method shows a complexity of $\tilde{O}(p^4 \sigma^2 + p^2 L)$, and thus only a partial improvement (even without considering the initial root isolation step). The reason is that we are not aware of an improved bound for the initial sequence of a single α compared to what we prove in Theorem 2.

From a theoretical point of view, we do not expect significant improvements when using any other quadratic convergent method than qir: the first summand $p^4 \sigma^2$ of Theorem 4 appears due to root isolation, and the summand $p^3 L$ seems to be unavoidable as well, since for each root, one has to perform at least one evaluation of f for a rational number of bitsize $O(L)$, which leads to $O(n)$ arithmetic operations with integers of bitsize up to $\tilde{O}(nL)$.

5 Conclusions and Further Work

Theorem 4 shows that refining all real roots of a polynomial to width ϵ is as complex as just isolating the roots, provided that $\log \epsilon^{-1} = \tilde{O}(p\sigma^2)$. We believe this result to be of general interest for algorithm dealing with real algebraic numbers. For instance, the usage of qir instead of naive bisection removes the asymptotic bottleneck in the topology computation algorithm presented in [11]; this is currently work in progress.

On the practical side, we have argued that qir has a more adaptive behavior than a combination of bisection and Newton's method, since the switch from linear to quadratic convergence happens without a "manual" control from outside. Abbott's work [1] has already shown that qir is competitive to Newton's method in a different context. However, a comparison to other hybrid approaches like Brent's method is still missing.

Acknowledgements. The author would like to thank Tobias Gärtner, Kurt Mehlhorn, Michael Sagraloff and Vikram Sharma for valuable discussions.

References

1. Abbott, J.: Quadratic Interval Refinement for Real Roots. Poster presented at the 2006 Int. Symp. on Symb. and Alg. Comp, ISSAC 2006 (2006), <http://www.dima.unige.it/~abbott/>
2. Basu, S., Pollack, R., Roy, M.-F.: Algorithms in Real Algebraic Geometry. In: Algorithms and Computation in Mathematics, 2nd edn., vol. 10. Springer, Heidelberg (2006)
3. Brent, R.: Algorithms for Minimization without Derivatives, ch. 4. Prentice-Hall, Englewood Cliffs (1973)
4. Caviness, B.F., Johnson, J.R. (eds.): Quantifier Elimination and Cylindrical Algebraic Decomposition, Texts and Monographs in Symbolic Computation. Springer, Heidelberg (1998)
5. Collins, G.E., Akritas, A.G.: Polynomial Real Root Isolation Using Descartes' Rule of Signs. In: Jenks, R.D. (ed.) SYMSAC, pp. 272–275. ACM Press, Yorktown Heights (1976)
6. Collins, G.E., Loos, R.: Real zeroes of polynomials. In: Computer algebra: symbolic and algebraic computation, 2nd edn., pp. 83–94. Springer, New York (1983)
7. Dekker, T.: Finding a zero by means of successive linear interpolation. In: Dejon, B., Henrici, P. (eds.) Constructive Aspects of the Fundamental Theorem of Algebra (1969)
8. Diochnos, D.I., Emiris, I.Z., Tsigaridas, E.P.: On the complexity of real solving bivariate systems. In: ISSAC 2007: Proceedings of the 2007 international symposium on Symbolic and algebraic computation, pp. 127–134. ACM, New York (2007)
9. Du, Z., Sharma, V., Yap, C.: Amortized Bound for Root Isolation via Sturm Sequences. In: Proceedings of the International Workshop on Symbolic-Numeric Computation, SNC 2007 (2007)
10. Eigenwillig, A., Kerber, M.: Exact and efficient 2D-arrangements of arbitrary algebraic curves. In: Proc. of the nineteenth annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008), pp. 122–131 (2008)
11. Eigenwillig, A., Kerber, M., Wolpert, N.: Fast and Exact Geometric Analysis of Real Algebraic Plane Curves. In: Brown, C.W. (ed.) Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation, ISSAC 2007 (2007)

12. Eigenwillig, A., Sharma, V., Yap, C.: Almost Tight Recursion Tree Bounds for the Descartes Method. In: Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation, pp. 71–78 (2006)
13. Emiris, I.Z., Hemmer, M., Karavelas, M., Mourrain, B., Tsigaridas, E.P., Zafeirakopoulos, Z.: Experimental evaluation and cross-benchmarking of univariate real solvers. Rapport de recherche EMIRIS:2008:INRIA-00340887:1, INRIA, Sophia Antipolis, France (2008)
14. Johnson, J.R.: Algorithms for Real Root Isolation. In: Caviness and Johnson [4] (1998)
15. Schönhage, A., Strassen, V.: Schnelle Multiplikation grosser Zahlen. Computing 7, 281–292 (1971)
16. Sharma, V.: Complexity of real root isolation using continued fractions. Theoretical Computer Science 409, 292–310 (2008)
17. Yap, C.K.: Fundamental Problems in Algorithmic Algebra. Oxford University Press, Oxford (2000)

Algebraic Approach to the Computation of the Defining Polynomial of the Algebraic Riccati Equation

Takuya Kitamoto

Faculty of Education, Yamaguchi University
kitamoto@yamaguchi-u.ac.jp

Abstract. The algebraic Riccati equation, which we denote by 'ARE' in the rest of the paper, is one of the most important equations of the post modern control theory. It plays important role for solving H_2 and H_∞ optimal control problems.

Although a well-known numerical algorithm can compute the solution of ARE efficiently ([1],[2]), the algorithm can not be applied when a given system contains an unknown parameter.

This paper presents an algorithm to compute the defining polynomial of an ARE with unknown parameter k . Such algorithm is also discussed in [3], where an algorithm with numerical approach is presented. The new algorithm in this paper uses algebraic approaches based on Groebner basis and resultant. Numerical experiments show the new algorithm is more efficient than that of [3] in most cases.

1 Introduction

Recently, Computer Algebra has received an increasing attention from engineers and scientists, because its capability to handle parameters can lead to a wide variety of applications. In the field of control theory, various techniques of Computer Algebra are proved to be quite effective for a design and analysis of control systems. For example, QE (Quantifier Elimination), which is a comparatively new technique in Computer Algebra, is applied the design of a control system in the references [4]-[7]. Another application of Computer Algebra to the control theory can be found in references [8] and [9], where the author of this paper described an algorithm to compute H_∞ norm of a given system that contains a parameter.

In this paper, we treat the 'parametric' algebraic Riccati equation, which is the algebraic Riccati equation containing an unknown parameter. The algebraic Riccati equation, which we denote by 'ARE' in the rest of the paper, is the equation in the form of

$$PA + A^T P - PWP + Q = 0, \quad (1)$$

where we are given $n \times n$ matrices A, W, Q (Q being positive semi definite and W being symmetric) and find the $n \times n$ solution matrix P . It is used for solving H_2

and H_∞ optimal control problems, and is one of the most important equations in control theory. Typically, the solution P of the equation is computed by the numerical algorithm, which utilizes the eigenvalues and eigenvectors of the matrix H defined by

$$H \stackrel{\text{def}}{=} \begin{bmatrix} A & -W \\ -Q & -A^T \end{bmatrix}. \tag{2}$$

However, when matrices A, W, Q contain unknown parameters, the numerical algorithm can not be applied.

Letting each entry of the solution matrix be unknown variables, ARE can be viewed as m simultaneous algebraic equations with m variables and a parameter k , where m is the number of entries of the unknown matrix P . Hence, computing Groebner basis of the algebraic equations with lexicographic ordering, we obtain the polynomial whose roots are the solution of ARE, which is the defining polynomial of ARE.

However, in practice, the procedure is effective only for quite small size problems because of its numerical complexities. Reference [3] presents a practical algorithm to compute the defining polynomial of the solution P . The algorithm uses polynomial interpolations and the numerical solutions of ARE (the algorithm to compute numerical solutions is already known in control theory). In this paper, we present another algorithm which uses Groebner basis and resultant. Numerical experiments show that the new algorithm is more efficient than that of [3] in most cases. In the rest of the paper, we use the following notations:

- R** : The set of real numbers.
- C** : The set of complex numbers.
- Z** : The set of integers.
- F_p** : Finite field with characteristic p .
- K^{m,n}** : The set of matrices with entries in **K**.
- K[x]** : The set of polynomials with coefficients in **K**.
- Res_x(r₁(x), r₂(x))** : Resultant of polynomials $r_1(x)$ and $r_2(x)$ with respect to x .
- GCD_x(r₁(x), ..., r_n(x))** : Polynomial GCD of $r_1(x), \dots, r_n(x)$ with respect to x .
- H_{|_{k=k₀}}** : H with $k = k_0$.
- Det(M)** : Determinant of matrix M .

2 Original Algorithm

2.1 Problem Formulations

Let $A \in \mathbf{Z}[k]^{n,n}$, $W \in \mathbf{Z}[k]^{n,n}$, $Q \in \mathbf{Z}[k]^{n,n}$ be polynomial matrices in k , where W is symmetric and Q is semi definite for any value of k . Let $p_{i,j}$ be (i, j) th entry of a symmetric solution P of (1). We will compute the defining polynomial of $p_{i,j}$. For the symmetric solutions, we have the following theorem:

Theorem 1. *Let $p_{i,j}$ be (i, j) th entry of a symmetric solution P of (1), and suppose that there exists real number k_0 such that*

- (C1) *All eigenvalues of $H|_{k=k_0}$ are distinct,*

(C2) $\lambda_i + \lambda_j = 0 \ (i \neq j) \Rightarrow P(\lambda_1, \dots, \lambda_n)$ is not symmetric, where λ_i, λ_j are eigenvalues of $H|_{k=k_0}$.

Then there exists a polynomial $f_l(k) \in \mathbf{Z}[k] \ (l = 0, \dots, 2^n)$ which satisfies the conditions

$$\text{GCD}_k(f_0(k), \dots, f_{2^n}(k)) = 1, \tag{3}$$

$$f_{2^n}(k)p_{i,j}^{2^n} + \dots + f_1(k)p_{i,j} + f_0(k) = 0. \tag{4}$$

In this paper, we assume the above conditions (C1) and (C2), and present an algorithm to compute the polynomial (4).

2.2 Algorithm with Numerical Approach

In this subsection, we explain the algorithm in [3] briefly, which computes the defining polynomial numerically. For details, see [3].

Let H be the matrix defined by (2), and compute vector $v(y)$ with the following algorithm:

Algorithm 1. Computation of $v(\lambda)$

- (1) Let x be $x = [x_1 \ \dots \ x_{2n}]^T$ and compose $2n$ linear equations $(H - \lambda E)x = 0$ (each entry of $(H - \lambda E)x = 0$ compose a linear equation), where λ is an indeterminate.
- (2) Select $(2n - 1)$ linear equations from $2n$ ones in (1) and solve the $(2n - 1)$ linear equations as equations in variables x_1, \dots, x_{2n-1} .
- (3) Substitute the solution of x_1, \dots, x_{2n-1} into x and multiply an adequate polynomial so that each entry of x is a polynomial in λ .
- (4) Let $v(\lambda) \leftarrow x/x_{2n}$ and output $v(\lambda)$.

From **Theorem 1**, we see that there exists 2^n symmetric solutions of ARE when $k = k_0$, provided that $f_{2^n}(k_0) \neq 0$. For the 2^n symmetric solutions, we have the following theorem:

Theorem 2. Let $\pm\lambda_1, \dots, \pm\lambda_n$ be the eigenvalues of H in (2). Then, (i, j) th entry of the 2^n symmetric solutions are given by

$$\frac{\text{Det}(\bar{\Gamma}_{i,j}(s_1\lambda_1, \dots, s_n\lambda_n))}{\text{Det}(\Gamma_1(s_1\lambda_1, \dots, s_n\lambda_n))} \quad (s_l = \pm 1, \ l = 1, \dots, n), \tag{5}$$

where $\Gamma_1(y_1, \dots, y_n)$ and $\bar{\Gamma}_{i,j}(y_1, \dots, y_n)$ are matrices defined by ($v_i(y)$ denotes i -th entry of vector $v(y)$ computed by **Algorithm 2**)

$$\Gamma_1(y_1, \dots, y_n) \stackrel{\text{def}}{=} \begin{bmatrix} v_1(y_1) & \dots & v_1(y_n) \\ \vdots & & \vdots \\ v_n(y_1) & \dots & v_n(y_n) \end{bmatrix}, \tag{6}$$

$$\bar{\Gamma}_{i,j}(y_1, \dots, y_n) \stackrel{\text{def}}{=} \begin{bmatrix} v_1(y_1) & \cdots & v_1(y_n) \\ \vdots & \vdots & \vdots \\ v_{j-1}(y_1) & \cdots & v_{j-1}(y_n) \\ v_{n+i}(y_1) & \cdots & v_{n+i}(y_n) \\ v_{j+1}(y_1) & \cdots & v_{j+1}(y_n) \\ \vdots & \vdots & \vdots \\ v_n(y_1) & \cdots & v_n(y_n) \end{bmatrix}. \tag{7}$$

From the definitions (6) and (7), we see that polynomials $\text{Det}(\Gamma_1(y_1, \dots, y_n))$ and $\text{Det}(\bar{\Gamma}_{i,j}(y_1, \dots, y_n))$ are alternative polynomials. Thus, there exist symmetric polynomials $g_1(y_1, \dots, y_n)$ and $\bar{g}_{i,j}(y_1, \dots, y_n)$ such that

$$\text{Det}(\Gamma_1(y_1, \dots, y_n)) = g_1(y_1, \dots, y_n) \prod_{s < t} (y_s - y_t), \tag{8}$$

$$\text{Det}(\bar{\Gamma}_{i,j}(y_1, \dots, y_n)) = \bar{g}_{i,j}(y_1, \dots, y_n) \prod_{s < t} (y_s - y_t). \tag{9}$$

Therefore, the above theorem implies that (i, j) th entry $p_{i,j}$ of the 2^n symmetric solutions are given by

$$\frac{\bar{g}_{i,j}(s_1 \lambda_1, \dots, s_n \lambda_n)}{g_1(s_1 \lambda_1, \dots, s_n \lambda_n)} \quad (s_l = \pm 1, l = 1, \dots, n), \tag{10}$$

and the defining polynomial of $p_{i,j}$ divides

$$\prod_{s_l = \pm 1} \{g_1(s_1 \lambda_1, \dots, s_n \lambda_n) p_{i,j} - \bar{g}_{i,j}(s_1 \lambda_1, \dots, s_n \lambda_n)\}. \tag{11}$$

This implies that head coefficient $f_{2^n}(k)$ of (4) divides

$$\phi(k) \stackrel{\text{def}}{=} \prod_{s_l = \pm 1} g_1(s_1 \lambda_1, \dots, s_n \lambda_n), \tag{12}$$

and there exists polynomial $\bar{f}(k) \in \mathbf{Z}[k]$ such that $\phi(k) = f_{2^n}(k) \bar{f}(k)$. Then, we can compute the polynomial in (4) as follows: Let k_r be an integer and let $\alpha_l(k_r)$ ($l = 1, \dots, 2^n$) denote 2^n roots of (4) when $k = k_r$ ($\alpha_l(k_r)$ can be computed by numerical algorithms in [1]). Since $\alpha_l(k_r)$ are (i, j) th entries of 2^n symmetric solutions of (1), we can compute them numerically with a conventional numerical method by substituting $k = k_r$. Hence, if we have $\phi(k)$ in (12), then we can compute

$$\begin{aligned} &\text{Eq. (11) with } k = k_r \\ &= \phi(k_r) \{(p_{i,j} - \alpha_1(k_r)) \cdots (p_{i,j} - \alpha_{2^n}(k_r))\} \\ &= f_{2^n}(k_r) \bar{f}(k_r) \{(p_{i,j} - \alpha_1(k_r)) \cdots (p_{i,j} - \alpha_{2^n}(k_r))\} \\ &= \bar{f}(k_r) \left\{ f_{2^n}(k_r) p_{i,j}^{2^n} + \cdots + f_1(k_r) p_{i,j} + f_0(k_r) \right\} \\ &= \sum_{l=0}^{2^n} \bar{f}(k_r) f_l(k_r) p_{i,j}^l \quad (\in \mathbf{Z}[p_{i,j}]). \end{aligned} \tag{13}$$

Looking at the coefficients of (13), we obtain $\bar{f}(k_r)f_l(k_r) \in \mathbf{Z}$ for any k_r , which implies that we can compute a polynomial $\bar{f}(k)f_l(k) \in \mathbf{Z}[k]$, using polynomial interpolation. With $\bar{f}(k)f_l(k)$ obtained, we compute $f_l(k)$ ($l = 0, \dots, 2^n$) by factoring $\sum_{l=0}^{2^n} \bar{f}(k)f_l(k)p_{i,j}^l$ as

$$\sum_{l=0}^{2^n} \bar{f}(k)f_l(k)p_{i,j}^l = \bar{f}(k) \left\{ f_{2^n}(k)p_{i,j}^{2^n} + \dots + f_1(k)p_{i,j} + f_0(k) \right\}, \quad (14)$$

from which we obtain polynomial (4). Thus, we obtain the following algorithm to compute (4).

Algorithm 2 (3). Computation of (4)

- (1) For $k = k_r \in \mathbf{Z}$ ($r = 1, \dots, m$) do the following:
 Let $\pm\lambda_1, \dots, \pm\lambda_n \in \mathbf{C}$ be the eigenvalues of $H|_{k=k_r}$ and compute

$$\phi(k_r) = \prod_{s_i=\pm 1} g_1(s_1\lambda_1, \dots, s_n\lambda_n) \ (\in \mathbf{Z}). \quad (15)$$

- (2) Compute $\phi(k)$ ($\in \mathbf{Z}[k]$) by polynomial interpolations.
- (3) For $k = k_r \in \mathbf{Z}$ ($r = 1, \dots, m$) do the following:
 Let $\alpha_1(k_r), \dots, \alpha_{2^n}(k_r) \in \mathbf{C}$ the (i, j) th entry of the solution of ARE, and compute $\bar{f}(k_r)f_l(k_r) \in \mathbf{Z}$ in (13).
- (4) Compute $\bar{f}(k)f_l(k) \in \mathbf{Z}[k]$ by polynomial interpolations. Then compute $f_l(k)$ by factorization (14).

3 Algebraic Algorithm

3.1 Setting

Let $g_1(y_1, \dots, y_n)$ and $\tilde{g}_{i,j}(y_1, \dots, y_n)$ be polynomials defined by (8) and (9). The previous section tells us that the defining polynomial can be computed from (14), provided that (11) can be computed as a univariate polynomial in $p_{i,j}$ for $k = k_r$ ($\in \mathbf{Z}$). Thus, first, we present algorithms to compute (11) with $k = k_r$ ($\in \mathbf{Z}$) as a univariate polynomial in $p_{i,j}$. Then, the polynomial in (14) can be computed by polynomial interpolations with respect to k .

3.2 Preliminaries

Let σ_r ($r = 1, \dots, n$) denote r -th elementary symmetric polynomials of $\lambda_1, \dots, \lambda_n$, i.e.

$$\sigma_1 = \lambda_1 + \dots + \lambda_n, \dots, \sigma_n = \lambda_1 \cdots \lambda_n, \quad (16)$$

where $\pm\lambda_1, \dots, \pm\lambda_n$ are the eigenvalues of H in (2). The characteristic polynomial of H can be expressed as

$$\begin{aligned} \text{Det}(xE - H) &= (x^2 - \lambda_1^2) \cdots (x^2 - \lambda_n^2) \\ &= h_0(k) + \dots + h_{2(n-1)}(k)x^{2(n-1)} + x^{2n}, \end{aligned} \quad (17)$$

where $h_{2r}(k)$ ($r = 0, \dots, 2(n-1)$) are polynomial in k satisfying

$$(-1)^n h_0(k) = \lambda_1^2 \cdots \lambda_n^2, \dots, (-1)h_{2(n-1)}(k) = \lambda_1^2 + \cdots + \lambda_n^2. \tag{18}$$

Obviously, right-hand sides of (18) are symmetric polynomials in $\lambda_1, \dots, \lambda_n$, and can be written as polynomials in $\sigma_1, \dots, \sigma_n$, i.e. there exist polynomials $\zeta_r(\sigma_1, \dots, \sigma_n)$ ($r = 0, \dots, n-1$) such that

$$(-1)^n h_0(k) = \zeta_0(\sigma_1, \dots, \sigma_n), \dots, (-1)h_{2(n-1)}(k) = \zeta_{n-1}(\sigma_1, \dots, \sigma_n). \tag{19}$$

When k is fixed to an integer $k_0 \in \mathbf{Z}$,

$$(-1)^n h_0(k_0) = \zeta_0(\sigma_1, \dots, \sigma_n), \dots, (-1)h_{2(n-1)}(k_0) = \zeta_{n-1}(\sigma_1, \dots, \sigma_n). \tag{20}$$

is a system of polynomials in $\sigma_1, \dots, \sigma_n$, whose solution is given by

$$\bar{\sigma}_1 = \bar{\lambda}_1 + \cdots + \bar{\lambda}_n, \dots, \bar{\sigma}_n = \bar{\lambda}_1 \cdots \bar{\lambda}_n, \tag{21}$$

where $\pm \bar{\lambda}_1, \dots, \pm \bar{\lambda}_n (\in \mathbf{C})$ are eigenvalues of $H|_{k=k_0}$. Since there are only finitely many eigenvalues of $H|_{k=k_0}$, there are only finitely many solutions $(\bar{\sigma}_1, \dots, \bar{\sigma}_n)$ of (20).

Let $\tau_t()$ ($t = 1, \dots, 2^n$) be a function which maps λ_r ($r = 1, \dots, n$) to $s_r \lambda_r$ ($s_r = \pm 1$). For example, when $n = 2$, $\tau_t()$ ($t = 1, \dots, 4$) can be defined by

$$\begin{aligned} \tau_1(h(\lambda_1, \lambda_2)) &= h(\lambda_1, \lambda_2), & \tau_2(h(\lambda_1, \lambda_2)) &= h(-\lambda_1, \lambda_2), \\ \tau_3(h(\lambda_1, \lambda_2)) &= h(\lambda_1, -\lambda_2), & \tau_4(h(\lambda_1, \lambda_2)) &= h(-\lambda_1, -\lambda_2). \end{aligned}$$

Similarly, we define $\bar{\tau}()$ as a function which maps $\bar{\lambda}_r$ to $s_r \bar{\lambda}_r$ ($s_r = \pm 1$). If $(\sigma_1, \dots, \sigma_n) = (\bar{\sigma}_1, \dots, \bar{\sigma}_n)$ is a solution of (20), then

$$(\sigma_1, \dots, \sigma_n) = (\tau_t(\bar{\sigma}_1), \dots, \tau_t(\bar{\sigma}_n)) \quad (t = 1, \dots, 2^n)$$

are also solutions of (20), since they satisfy

$$\begin{aligned} \zeta_0(\bar{\tau}_t(\bar{\sigma}_1), \dots, \bar{\tau}_t(\bar{\sigma}_n)) &= \bar{\lambda}_1^2 \cdots \bar{\lambda}_n^2 = (-1)^n h_0(k_0), \\ &\vdots \\ \zeta_{n-1}(\bar{\tau}_t(\bar{\sigma}_1), \dots, \bar{\tau}_t(\bar{\sigma}_n)) &= \bar{\lambda}_1^2 + \cdots + \bar{\lambda}_n^2 = (-1)h_{2(n-1)}(k_0). \end{aligned}$$

Thus, we have the following lemma:

Lemma 1. *There are only finitely many solutions of $\sigma_1, \dots, \sigma_n$ of (20). More concretely, there are 2^n solutions of $\sigma_1, \dots, \sigma_n$ with multiplicities counted.*

Let θ be defined by

$$\theta = z_1 \sigma_1 + \cdots + z_n \sigma_n, \tag{22}$$

and consider a system of polynomial equations

$$\begin{aligned} \theta &= z_1 \sigma_1 + \cdots + z_n \sigma_n, & \zeta_0(\sigma_1, \dots, \sigma_n) &= (-1)^n h_0(k_0), \\ &\dots, & \zeta_{n-1}(\sigma_1, \dots, \sigma_n) &= -h_{2(n-1)}(k_0), \end{aligned} \tag{23}$$

where z_1, \dots, z_n are some integers. If $(\sigma_1, \dots, \sigma_n) = (\bar{\sigma}_1, \dots, \bar{\sigma}_n)$ is a solution of (20), then

$$\theta = z_1 \bar{\tau}_t(\bar{\sigma}_1) + \dots + z_n \bar{\tau}_t(\bar{\sigma}_n)$$

are solutions of (23) with respect to θ . Hence, the defining polynomial of θ is given by

$$\prod_{i=1}^{2^n} \{\theta - (z_1 \bar{\tau}_t(\bar{\sigma}_1) + \dots + z_n \bar{\tau}_t(\bar{\sigma}_n))\}. \tag{24}$$

Since there are only finitely many solutions of (19) with $k = k_0$ ($k_0 \in \mathbf{Z}$), so are the solutions of (23). Therefore, there are integers z_1, \dots, z_n such that Groebner basis of

$$\{z_1 \sigma_1 + \dots + z_n \sigma_n - \theta, \zeta_0(\sigma_1, \dots, \sigma_n) - (-1)^n h_0(k_0), \dots, \zeta_{n-1}(\sigma_1, \dots, \sigma_n) + h_{2(n-1)}(k_0)\} \tag{25}$$

with lexicographic ordering $\sigma_1, \dots, \sigma_n \succ \theta$ is in the form of in shape basis $\{w_0(\theta), w_1(\theta, \sigma_1), \dots, w_n(\theta, \sigma_n)\}$, where $w_0(\theta)$ and $w_r(\theta, \sigma_r)$ are polynomials in the form of

$$w_0(\theta) = \theta^{2^n} + w_{0,2^n-1} \theta^{2^n-1} + \dots + w_{0,0},$$

$$w_r(\theta, \sigma_r) = w_{r,2^n} \sigma_r + w_{r,2^n-1} \theta^{2^n-1} + \dots + w_{r,1} \theta + w_{r,0} \quad (r = 1, \dots, n) \tag{26}$$

with $w_{r,t}, w_{r,2^n} \in \mathbf{Z}$ ($r = 1, \dots, n, t = 0, \dots, 2^n - 1$) (we note that $w_0(\theta)$ is monic, since we have $w_0(\theta) = (24)$). In fact, for almost all integers z_1, \dots, z_n and k_0 , Groebner basis of (25) is in the form of shape basis.

Since $g_1(y_1, \dots, y_n)$ and $\bar{g}_{i,j}(y_1, \dots, y_n)$ in (8), (9) are symmetric polynomials in y_1, \dots, y_n , there exist polynomials $u_1(\sigma_1, \dots, \sigma_n)$ and $\bar{u}_{i,j}(\sigma_1, \dots, \sigma_n)$ such that

$$u_1(\sigma_1, \dots, \sigma_n) = g_1(\lambda_1, \dots, \lambda_n), \quad \bar{u}_{i,j}(\sigma_1, \dots, \sigma_n) = \bar{g}_{i,j}(\lambda_1, \dots, \lambda_n). \tag{27}$$

From the definition of $\tau_t()$, we have an integer t ($1 \leq t \leq 2^n$) such that

$$\begin{aligned} \tau_t(u_1(\sigma_1, \dots, \sigma_n) p_{i,j} - \bar{u}_{i,j}(\sigma_1, \dots, \sigma_n)) \\ = \tau_t(g_1(\lambda_1, \dots, \lambda_n) p_{i,j} - \bar{g}_{i,j}(\lambda_1, \dots, \lambda_n)) \\ = g_1(s_1 \lambda_1, \dots, s_n \lambda_n) p_{i,j} - \bar{g}_{i,j}(s_1 \lambda_1, \dots, s_n \lambda_n) \end{aligned} \tag{28}$$

for any $s_l (= \pm 1)$ ($l = 1, \dots, n$). Hence, we have

$$\begin{aligned} \prod_{s_l = \pm 1} \{g_1(s_1 \lambda_1, \dots, s_n \lambda_n) p_{i,j} - \bar{g}_{i,j}(s_1 \lambda_1, \dots, s_n \lambda_n)\} = \\ \prod_{t=1}^{2^n} \tau_t(u_1(\sigma_1, \dots, \sigma_n) p_{i,j} - \bar{u}_{i,j}(\sigma_1, \dots, \sigma_n)). \end{aligned} \tag{29}$$

Let $\xi_r(\theta) (\in \mathbf{Q}[\theta])$ be defined by

$$\xi_r(\theta) \stackrel{\text{def}}{=} - \left(\frac{w_{r,2^n-1} \theta^{2^n-1} + \dots + w_{r,1} \theta + w_{r,0}}{w_{r,2^n}} \right), \tag{30}$$

where $w_{r,2^n}, w_{r,t}$ are integers given by (26). From (26), we see that

$$\sigma_r = -\frac{w_{r,2^n-1}\theta^{2^n-1} + \dots + w_{r,1}\theta + w_{r,0}}{w_{r,2^n}} = \xi_r(\theta), \tag{31}$$

which implies that

$$\begin{aligned} &\tau_t(u_1(\sigma_1, \dots, \sigma_n)p_{i,j} - \bar{u}_{i,j}(\sigma_1, \dots, \sigma_n)) = \\ &\tau_t(u_1(\xi_1(\theta), \dots, \xi_n(\theta))p_{i,j} - \bar{u}_{i,j}(\xi_1(\theta), \dots, \xi_n(\theta))). \end{aligned} \tag{32}$$

The following algorithm is the key of our algorithm.

Theorem 3. *Let k_0, z_1, \dots, z_n be integers such that Groebner basis of (25) is in the form of shape basis (26). Then, we have*

$$\begin{aligned} &\prod_{s_i=\pm 1} \{g_1(s_1\bar{\lambda}_1, \dots, s_n\bar{\lambda}_n)p_{i,j} - \tilde{g}_{i,j}(s_1\bar{\lambda}_1, \dots, s_n\bar{\lambda}_n)\} \\ &= \text{Res}_\theta(w_0(\theta), \Phi(\theta, p_{i,j})), \end{aligned} \tag{33}$$

where $\pm\bar{\lambda}_1, \dots, \pm\bar{\lambda}_n \in \mathbf{C}$ are the eigenvalues of $H|_{k=k_0}$, $\Phi(\theta, p_{i,j}) (\in \mathbf{Q}[\theta, p_{i,j}])$ is a polynomial defined by

$$\Phi(\theta, p_{i,j}) \stackrel{\text{def}}{=} u_1(\xi_1(\theta), \dots, \xi_n(\theta))p_{i,j} - \bar{u}_{i,j}(\xi_1(\theta), \dots, \xi_n(\theta)), \tag{34}$$

and $\xi_r(\theta) (r = 1, \dots, n)$, $u_1(\sigma_1, \dots, \sigma_n)$, $\bar{u}_{i,j}(\sigma_1, \dots, \sigma_n)$ are polynomials defined by (30), (27).

Proof

Let $\bar{\theta} (\in \mathbf{C})$ be defined by

$$\bar{\theta} = z_1\bar{\sigma}_1 + \dots + z_n\bar{\sigma}_n, \tag{35}$$

where $\bar{\sigma}_1, \dots, \bar{\sigma}_n$ are given by (21). Since

$$(\sigma_1, \dots, \sigma_n) = (\bar{\tau}_t(\bar{\sigma}_1), \dots, \bar{\tau}_t(\bar{\sigma}_n)) \quad (1 \leq t \leq 2^n)$$

is a solution of (25),

$$\bar{\tau}_t(\bar{\theta}) = z_1\bar{\tau}_t(\bar{\sigma}_1) + \dots + z_n\bar{\tau}_t(\bar{\sigma}_n) \quad (t = 1, \dots, 2^n) \tag{36}$$

are the roots of $w_0(\theta)$ in (26). Hence, from the property of the resultant, we see

$$\text{Res}_\theta(w_0(\theta), \Phi(\theta, p_{i,j})) = \prod_{t=1}^{2^n} \Phi(\bar{\tau}_t(\bar{\theta}), p_{i,j}). \tag{37}$$

Therefore, we obtain

$$\prod_{s_i=\pm 1} \{g_1(s_1\bar{\lambda}_1, \dots, s_n\bar{\lambda}_n)p_{i,j} - \bar{g}_{i,j}(s_1\bar{\lambda}_1, \dots, s_n\bar{\lambda}_n)\}$$

$$\begin{aligned}
 &= \prod_{t=1}^{2^n} \bar{\tau}_t (u_1(\bar{\sigma}_1, \dots, \bar{\sigma}_n) p_{i,j} - \bar{u}_{i,j}(\bar{\sigma}_1, \dots, \bar{\sigma}_n) p_{i,j}) \quad (\because \text{\textcircled{29}}) \\
 &= \prod_{t=1}^{2^n} \bar{\tau}_t (u_1(\xi_1(\bar{\theta}), \dots, \xi_n(\bar{\theta})) p_{i,j} - \bar{u}_{i,j}(\xi_1(\bar{\theta}), \dots, \xi_n(\bar{\theta}))) \quad (\because \text{\textcircled{32}}) \\
 &= \prod_{t=1}^{2^n} \bar{\tau}_t (\Phi(\bar{\theta}, p_{i,j})) \quad (\because \text{\textcircled{34}}) \\
 &= \prod_{t=1}^{2^n} \Phi(\bar{\tau}_t(\bar{\theta}), p_{i,j}) = \text{Res}_\theta(w_0(\theta), \Phi(\theta, p_{i,j})) \quad (\because \text{\textcircled{37}}), \tag{38}
 \end{aligned}$$

which proves the theorem. □

3.3 Algorithm Description

From **Theorem 3**, we obtain the following algorithm to compute the value of **\text{\textcircled{11}}**.

Algorithm 3. Computation of **\text{\textcircled{11}} for $k = k_0$**

Input: $k_0 \in \mathbf{Z}$ and ARE in **\text{\textcircled{1}}** with parameter k .

Output: **\text{\textcircled{11}}** where $\pm\lambda_1, \dots, \pm\lambda_n$ are eigenvalues of $H|_{k=k_0}$.

- (1) Compute $u_1(\sigma_1, \dots, \sigma_n), \bar{u}_{i,j}(\sigma_1, \dots, \sigma_n)$ in **\text{\textcircled{27}}**.
- (2) Let $\zeta_0(\sigma_1, \dots, \sigma_n), \dots, \zeta_{n-1}(\sigma_1, \dots, \sigma_n)$ be polynomial in $\sigma_1, \dots, \sigma_n$ such that

$$\zeta_0(\sigma_1, \dots, \sigma_n) = \lambda_1^2 \cdots \lambda_n^2, \dots, \zeta_{n-1}(\sigma_1, \dots, \sigma_n) = \lambda_1^2 + \cdots + \lambda_n^2,$$

where $\sigma_1, \dots, \sigma_n$ are defined by **\text{\textcircled{16}}**.

- (3) Let z_1, \dots, z_n be some integers.
- (4) Compute Groebner basis of **\text{\textcircled{25}}**.
- (5) If Groebner basis computed in step (4) is not in the form of shape basis **\text{\textcircled{26}}**, then change integers z_1, \dots, z_n and go back to step (4).
- (6) Compute $\Phi(\theta, p_{i,j})$ in **\text{\textcircled{34}}**.
- (7) Compute $\text{Res}_\theta(w_0(\theta), \Phi(\theta, p_{i,j})) \in \mathbf{Z}[p_{i,j}]$, and output the polynomial.

Step (1) of **Algorithm 3** can be performed the algorithm in **\text{\textcircled{12}}**, where an efficient algorithm to compute the determinant of generalized Vandermonde Matrix

$$\begin{bmatrix} \psi_1(x_1) & \psi_2(x_1) & \cdots & \psi_n(x_1) \\ \psi_1(x_2) & \psi_2(x_2) & \cdots & \psi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(x_n) & \psi_2(x_n) & \cdots & \psi_n(x_n) \end{bmatrix} \quad (\psi_i(x_j) \in \mathbf{Z}[x_j])$$

is presented.

Step ⟨2⟩ of **Algorithm 3** can be performed by computing Groebner basis of

$$\{\mu - \Omega_r(\lambda_1^2, \dots, \lambda_n^2), \lambda_1 + \dots + \lambda_n - \sigma_1, \dots, \lambda_1 \cdots \lambda_n - \sigma_n\},$$

with lexicographic ordering $\lambda_1, \dots, \lambda_n \succ \sigma_1, \dots, \sigma_n, \mu$, where $\Omega_r(\lambda_1^2, \dots, \lambda_n^2)$ is the r -th symmetric polynomial of $\lambda_1^2, \dots, \lambda_n^2$. Thus, this step is independent of given input, and its computation can be performed beforehand.

Step ⟨3⟩-⟨5⟩ of **Algorithm 3** requires a computation of Groebner basis of (25) whose variety is zero dimensional, and can be performed efficiently.

Step ⟨6⟩ requires only substitution and can be performed efficiently, and Step ⟨7⟩ requires one resultant computation.

Thus, **Algorithm 3** requires the following computations:

- Computation of $u_1(\sigma_1, \dots, \sigma_n)$ and $\bar{u}_{i,j}(\sigma_1, \dots, \sigma_n)$ of (27) in step ⟨1⟩ (the algorithm in (12) can be used for this computation).
- Groebner basis of (25) in step ⟨3⟩.
- Resultant $\text{Res}_\theta(w_0(\theta), \Phi(\theta, p_{i,j}))$ in step ⟨7⟩.

Based on **Algorithm 3**, we present the following algorithm to compute the defining polynomial (4):

Algorithm 4. Computation of (4)

- ⟨1⟩ For $k = k_r \in \mathbf{Z}$ ($r = 1, \dots, m$) do the following: Compute (11) for $k = k_r$ with **Algorithm 3**. Then, compute $\bar{f}(k_r)f_l(k_r) \in \mathbf{Z}$ in (13).
- ⟨2⟩ Compute $\bar{f}(k)f_l(k) \in \mathbf{Z}[k]$ by polynomial interpolations. Then compute $f_l(k)$ by factorization (14).

4 Experiments

4.1 Setting

Let $A \in \mathbf{Z}[k]^{n,n}$, $B \in \mathbf{Z}^{1,n}$ be the following matrices:

$$A = k\bar{E} + \Omega_{n,n}, \quad B = \Omega_{1,n}, \tag{39}$$

where matrix $\Omega_{n,n} \in \mathbf{Z}^{n,n}$ is a randomly generated $n \times n$ matrix whose entries are integers between -5 and 5 , and \bar{E} is a matrix whose (i, j) -th entry $\bar{e}_{i,j}$ is defined by

$$\bar{e}_{i,j} = \begin{cases} \tau, & \text{when } (i, j) = (1, 1) \\ 0, & \text{otherwise} \end{cases}, \quad \tau = \text{random integer } (\neq 0) \text{ between } -5 \text{ and } 5.$$

For example, when $n = 2$, an example of A, B in (39) is

$$A = \begin{bmatrix} 2k - 3 & 3 \\ -1 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Numerical experiments are performed as follows: Matrices Q and R are set to the identity matrices, and we generate 5 sets of A, B for each $n = 2, 3, 4$. Then the defining polynomial (4) of symmetric solutions of ARE (1) is computed by the following two methods:

Table 1. Computation time (in seconds)

n	2	3	4
M1	1.406	20.61	445.9
M2	1.703	13.57	326.2

- (M1) The method in [3] (Algorithm 1 and 2).
- (M2) The method in this paper (Algorithm 1, 3 and 4).

The experiments are performed with Maple 11 on the machine equipped with Pentium M 2.0GHz and 1.5GByte memory.

4.2 Results

Table 1 shows the average of 5 computation times in seconds. From the table, we see that our new algorithm (Algorithm 4) is more efficient than that of [3] except for $n = 2$. Looking at the computation time closely, we find that almost all computation time is spent on step <7> (resultant computation). Hence, efficient resultant computation is the key to obtain further improvements of the efficiency of the algorithm.

5 Applications

The application of the algorithm in this paper is not limited to the computation of the defining polynomial of ARE. One of the application is [10], where it is shown that the minimum of cost function $\int_0^\infty (x^T Qx + u^T Ru) dt$ is a root of

$$\prod_{s_i = \pm 1} \{p_d(s_1 \lambda_1, \dots, s_n \lambda_n)q - p_n(s_1 \lambda_1, \dots, s_n \lambda_n)\}, \tag{40}$$

where $p_d(y_1, \dots, y_n)$ and $p_n(y_1, \dots, y_n)$ are certain polynomials symmetric in y_1, \dots, y_n , and $\pm \lambda_1, \dots, \pm \lambda_n$ are eigenvalues of H in (2) with $W = BR^{-1}B^T$. Obviously, given polynomials $p_d(y_1, \dots, y_n)$, $p_n(y_1, \dots, y_n)$ and matrix H , our algorithm can compute (40) as a polynomial in q . Another application is given in [11], where the optimal H_∞ norm achievable using a static feedback controller is expressed as a root of a polynomial. In the algorithm of [11], we need to compute $\prod_{s_i = \pm 1} \eta_1(s_1 \lambda_1, \dots, s_n \lambda_n)$, where $\eta_1(y_1, \dots, y_n)$ is a certain polynomial symmetric in y_1, \dots, y_n , and $\pm \lambda_1, \dots, \pm \lambda_n$ are eigenvalues of H in (2) with $W = B_2 B_2^T - \frac{1}{\gamma^2} B_1 B_1^T$, $Q = C^T C$ (B_1, B_2, C are certain matrices).

6 Conclusion

We present a new algorithm (Algorithm 4) to compute the defining polynomial of ARE, which produces the same results as the one in [3]. From the numerical

experiments, we see that the new algorithm is more efficient than that of [3] in most cases.

Applications of the algorithm in the paper is not limited to the computation of the defining polynomial of ARE. It can be used to the algorithm which computes (1) the minimum of the cost function $\int_0^\infty (x^T Qx + u^T Ru) dt$, (2) the optimal H_∞ norm achievable using a static feedback controller, as a root of a polynomial.

Our current targets are twofold: One is further improvement the efficiency of the algorithm in the paper, and the other is further extensions of its applications. Close examination of the numerical experiments reveals that the efficient resultant computation is the key for the further improvements of the efficiency of the algorithm.

References

1. Zhou, K., Doyle, J., Glover, K.: Robust and Optimal Control. Prentice-Hall. Inc., New Jersey (1996)
2. Nishimura, T., Kano, H.: Matrix Riccati Equations in Control Theory (in Japanese). Asakura-syoten, Tokyo (1996)
3. Kitamoto, T., Yamaguchi, T.: On the computation of the defining polynomial of the algebraic Riccati equation. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2007. LNCS, vol. 4770, pp. 224–235. Springer, Heidelberg (2007)
4. Abdallah, C., Dorato, P., Yang, W., Liska, R., Steinberg, S.: Application of Quantifier Elimination Theory to Control System Design. In: Proc. of 4th IEEE Mediterranean Symposium of Control and Automation, Maleme, Crete, pp. 340–345 (1996)
5. Anai, H., Yanami, H.: syNRAC: A maple-package for solving real algebraic constraints. In: Sloot, P.M.A., Abramson, D., Bogdanov, A.V., Gorbachev, Y.E., Dongarra, J., Zomaya, A.Y. (eds.) ICCS 2003. LNCS, vol. 2657, pp. 828–837. Springer, Heidelberg (2003)
6. Dorato, P., Yang, W., Abdallah, C.: Robust Multi-Objective Feedback Design by Quantifier Elimination. *J. Symbolic Computation* 24, 153–159 (1997)
7. Hong, H., Liska, R., Steinberg, S.: Testing Stability by Quantifier Elimination. *J. Symbolic Computation* 24, 161–187 (1997)
8. Kitamoto, T.: On the computation of H_∞ norm of a system with a parameter. *The IEICE Transaction on Fundamentals (Japanese edition)* J89-A(1), 25–39 (2006)
9. Kitamoto, T., Yamaguchi, T.: Parametric Computation of H_∞ Norm of a System. In: Proc. SICE-ICCAS 2006, Busan, Korea (2006)
10. Kitamoto, T., Yamaguchi, T.: On the parametric LQ control problem (in Japanese). *The IEICE Transaction on Fundamentals (Japanese edition)* 91(3), 349–359 (2008)
11. Kitamoto, T., Yamaguchi, T.: The Optimal H_∞ Norm of a Parametric System Achievable Using Static Feedback Controller. *IEICE Transaction on Fundamentals* E90-A(11), 2496–2509 (2007)
12. Kitamoto, T.: On the Computation of the Determinant of a Generalized Vandermonde Matrix. Submitted to *IEICE Transaction on Fundamentals*

Discrete Dynamics: Gauge Invariance and Quantization

Vladimir V. Kornyak

Laboratory of Information Technologies
Joint Institute for Nuclear Research
141980 Dubna, Russia
kornyak@jinr.ru

Abstract. Gauge invariance in discrete dynamical systems and its connection with quantization are considered. For a complete description of gauge symmetries of a system we construct explicitly a class of groups unifying in a natural way the *space* and *internal* symmetries. We describe the main features of the gauge principle relevant to the discrete and finite background. Assuming that continuous phenomena are approximations of more fundamental discrete processes, we discuss – with the help of a simple illustration – relations between such processes and their continuous approximations. We propose an approach to introduce quantum structures in discrete systems based on finite gauge groups. In this approach quantization can be interpreted as introduction of gauge connection of a special kind. We illustrate our approach to quantization by a simple model and suggest generalization of this model. One of the main tools for our study is a program written in C.

1 Introduction

In 1918 Hermann Weyl – guided by the concept that the scale of length is arbitrary: if there is no fundamental length in Nature it does not matter what unit of length is used in measurements – conjectured that the scale can be taken in the form $e^{S(x)}$, i.e., it may vary from point to point in time and space. This idea failed in application to physics but gave rise to the concept of *gauge invariance*.

Later – in 1929, after advent of Quantum Mechanics – Weyl (and also Vladimir Fock and Fritz London) replaced scale transformations $e^{S(x)}$ by rotations (phase transformations) $e^{iS(x)}$ and derived electromagnetism from the gauge principle.

In 1954 C.N. Yang and R. Mills extended the gauge principle to non-Abelian symmetries. Now the gauge principle is recognized as one of the central principles in contemporary physics – in fact, all fundamental physical theories are gauge theories (for historical review see [1]).

The lattice gauge theory was introduced by K.G. Wilson in 1974 as a practical approach to the problems of strong interactions for which the standard perturbative methods are inapplicable. This technique – based on approximation of space, or space-time, by some (usually hypercubic) lattice – was considered as an auxiliary computational method rather than a fundamental construction. The

later mathematical generalizations established relations between lattice gauge theories and such topics as *topological quantum field theory (TQFT)*, *invariants of 3- and 4-manifolds*, *monoidal categories*, *Hopf algebras and quantum groups*, *quantum gravity* etc., [2].

In view of their origin and applications, the above mentioned lattice gauge theories are not entirely discrete constructions. They involve continuous ingredients: gauge groups are Lie groups, Lagrangians and observables are real or complex functions. Furthermore, the gauge groups of these theories are groups of internal symmetries and do not involve the lattice symmetries. It seems desirable to include the space symmetries into construction of gauge group, since: (a) the quantum statistics of particles is characterized by the rules describing their behavior under permutations of the points of space; (b) there exist gauge theories that deduce gravity by interpreting the space or space-time symmetries as gauge groups.

In this paper we consider more radical version of discrete gauge invariance. All our manipulations including quantization remain within the framework of exact discrete mathematics requiring no more than the ring of algebraic integers (and sometimes the quotient field of this ring). Our study was carried out with the help of a program in C we are developing now.

2 Discrete Dynamics

We consider evolution in the discrete time $t \in \mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$.

Let the space X be a finite set of points: $X = \{x_1, \dots, x_{N_X}\}$. This – primordially amorphous – set may possess some structure: some points may be “closer” to each other than others. A mathematical abstraction of such a structure is an *abstract simplicial complex* – a collection of subsets of X (*simplices*) such that any subset of a simplex is also simplex. One-dimensional complexes, i.e., *graphs* (or *lattices*), are sufficient to formulate a gauge theory. The symmetry group of the space X is the graph automorphism group $G = \text{Aut}(X)$.

Table [1] shows some lattices with their symmetries. We use these lattices in our computer experiments. In the table, N_X and N_E are numbers of points and vertices in space X ; the *trivial*, *symmetric*, *cyclic*, *dihedral* and *alternating* groups are denoted by 1, $\text{Sym}(n)$, \mathbb{Z}_n , D_{2n} and $\text{Alt}(n)$, respectively; the signs \times and \rtimes denote *direct* and *semidirect* products, respectively. Note that the lattice denoted as *Toric square* $n \times n$ in the table has three times larger symmetry group at $n = 4$ than the general case formula predicts [1].

Let each point $x \in X$ take values in some finite *set of local states* $\Sigma = \{\sigma_1, \dots, \sigma_{N_\Sigma}\}$ possessing some symmetry group $\Gamma \leq \text{Sym}(\Sigma)$. Such groups are analogs of the “groups of internal symmetries” responsible for interactions in physical gauge theories. The state of a system as a whole is a function $\sigma(x) \in \Sigma^X$.

¹ N. Vavilov pointed out to the author that this extra symmetry can be explained by








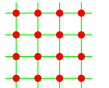

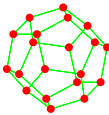
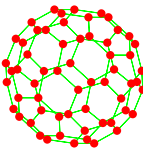
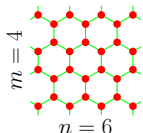
\mathbb{Z}_3 symmetry of the Dynkin diagram $D_4 =$  associated with the case $n = 4$.

Table 1. Examples of discrete spaces

X	N_X	N_E	G	$ G $
<i>Atom</i> 	1	0	1	1
<i>Dimer</i> 	2	1	$\text{Sym}(2) \cong \mathbb{Z}_2$	2
<i>Triangle</i> 	3	3	$\text{Sym}(3) \cong D_6$	6
<i>n-vertex polygon</i>	n	n	D_{2n}	$2n$
<i>Tetrahedron</i> 	4	6	$\text{Sym}(4)$	24
<i>Octahedron</i> 	6	12	$\mathbb{Z}_2 \times \text{Sym}(4)$	48
<i>Hexahedron</i> 	8	12	$\mathbb{Z}_2 \times \text{Sym}(4)$	48
<i>Toric square $n \times n, n \neq 4$</i> 	n^2	$2n^2$	$(\mathbb{Z}_n \times \mathbb{Z}_n) \rtimes D_8$	$8n^2$
$n = 4$	16	32	$((((\mathbb{Z}_2 \times D_8) \times \mathbb{Z}_2) \times \mathbb{Z}_3) \times \mathbb{Z}_2) \times \mathbb{Z}_2$	384
<i>Icosahedron</i> 	12	30	$\mathbb{Z}_2 \times \text{Alt}(5)$	120
<i>Dodecahedron</i> 	20	30	$\mathbb{Z}_2 \times \text{Alt}(5)$	120
<i>Fullerene C_{60}</i> 	60	90	$\mathbb{Z}_2 \times \text{Alt}(5)$	120
<i>Toric graphene $n \times m$</i>  $m = 4$ $n = 6$	nm	$\frac{3nm}{2}$	$D_n \times D_{2m}$	$2nm$

Dynamics of the system is determined by some *evolution rule* connecting the current state of the system $\sigma_t(x)$ with its prehistory $\sigma_{t-1}(x), \sigma_{t-2}(x), \dots$. A typical form of evolution rule is *evolution relation*:

$$R(\sigma_t(x); \sigma_{t-1}(x), \sigma_{t-2}(x), \dots) \subseteq \Sigma^X \times \Sigma^X \times \dots \quad (1)$$

Most commonly used in applications and convenient for study are *deterministic* (or *causal*) *dynamical systems*. The current state of deterministic system is uniquely determined by its prehistory, i.e., relations like (1) are *functional* and can be written in the form

$$\sigma_t(x) = F(\sigma_{t-1}(x), \sigma_{t-2}(x), \dots) \quad .$$

There are two important special types of *non-deterministic* dynamical systems:

- *lattice models in statistical mechanics* – special instances of Markov chains;
- *discrete quantum systems* obtained from classical systems by identification of their states with basis elements of complex Hilbert spaces.

For these systems, transition from one state to any other is possible with some probability controlled by additional structures: real (for Markov chains) or complex (for quantum systems) weights assigned to state transitions. In this paper we restrict our attention to the case of discrete quantum systems.

3 Unification of Space and Internal Symmetries

Having the groups G and Γ acting on X and Σ , respectively, we can combine them into a single group W which acts on the states Σ^X of the whole system. The group W can be identified, as a set, with the *Cartesian product* $\Gamma^X \otimes G$, where Γ^X is the set of Γ -valued functions on X . That is, every element $u \in W$ can be represented in the form $u = (\alpha(x), a)$, where $\alpha(x) \in \Gamma^X$ and $a \in G$.

In physics, it is usually assumed that the space and internal symmetries are independent, i.e., W is the *direct product* $\Gamma^X \times G$ with action² on Σ^X and multiplication rule:

$$\begin{aligned} \sigma(x)(\alpha(x), a) &= \sigma(x)\alpha(x) && \text{action} \quad , \\ (\alpha(x), a) * (\beta(x), b) &= (\alpha(x)\beta(x), ab) && \text{multiplication} \quad . \end{aligned} \quad (2)$$

Another standard construction is the *wreath product* $\Gamma \wr_X G$ having a structure of the semidirect product $\Gamma^X \rtimes G$ with action and multiplication

$$\begin{aligned} \sigma(x)(\alpha(x), a) &= \sigma(xa^{-1})\alpha(xa^{-1}) \quad , \\ (\alpha(x), a) * (\beta(x), b) &= (\alpha(x)\beta(xa), ab) \quad . \end{aligned} \quad (3)$$

² We write group actions *on the right*. This, more intuitive, convention is adopted in both GAP and MAGMA – the most widespread computer algebra systems with advanced facilities for computational group theory.

These examples are generalized by the following **Statement**:

There are equivalence classes of *split group extensions* $1 \rightarrow \Gamma^X \rightarrow W \rightarrow G \rightarrow 1$ determined by *antihomomorphisms* $\mu : G \rightarrow G$. The equivalence is described by *arbitrary* function $\kappa : G \rightarrow G$. The explicit formulas for main group operations — **action** on Σ^X , **multiplication** and **inversion** — are

$$\sigma(x)(\alpha(x), a) = \sigma(x\mu(a))\alpha(x\kappa(a)) \quad , \quad (4)$$

$$(\alpha(x), a) * (\beta(x), b) = (\alpha(x\kappa(ab)^{-1}\mu(b)\kappa(a))\beta(x\kappa(ab)^{-1}\kappa(b)), ab) \quad , \quad (5)$$

$$(\alpha(x), a)^{-1} = \left(\alpha(x\kappa(a^{-1})^{-1}\mu(a)^{-1}\kappa(a))^{-1}, a^{-1} \right) \quad . \quad (6)$$

This statement follows from the general description of the structure of split extensions of a group G_1 by a group G_0 : all such extensions are determined by the homomorphisms from G_1 to $\text{Aut}(G_0)$ (see, e.g., [3], p. 18). Specializing this description to the case when G_0 is the set of Γ -valued function on X and G_1 acts on arguments of these functions we obtain our statement. The *equivalence* of extensions with the same antihomomorphism μ but with different functions κ is expressed by the commutative diagram

$$\begin{array}{ccccccccc} 1 & \longrightarrow & \Gamma^X & \longrightarrow & W & \longrightarrow & G & \longrightarrow & 1 \\ & & \parallel & & \downarrow K & & \parallel & & \\ 1 & \longrightarrow & \Gamma^X & \longrightarrow & W' & \longrightarrow & G & \longrightarrow & 1 \end{array} \quad , \quad (7)$$

where the mapping K takes the form $K : (\alpha(x), a) \mapsto (\alpha(x\kappa(a)), a)$.

Note that the standard direct [2] and wreath [3] products are obtained from this general construction by choosing $(\mu(a) = 1, \kappa(a) = 1)$ and $(\mu(a) = a^{-1}, \kappa(a) = a^{-1})$, respectively.

In our C program the group W is specified by two groups G and Γ and two functions $\mu(a)$ and $\kappa(a)$ implemented as arrays. It is convenient in computations to use the following specialization: $\mu(a) = a^{-m}$ and $\kappa(a) = a^k$. For such a choice, formulas [4-6] take the form

$$\sigma(x)(\alpha(x), a) = \sigma(xa^{-m})\alpha(xa^k) \quad , \quad (8)$$

$$(\alpha(x), a) * (\beta(x), b) = (\alpha(x(ab)^{-k-m}a^{k+m})\beta(x(ab)^{-k}b^k), ab) \quad , \quad (9)$$

$$(\alpha(x), a)^{-1} = \left(\alpha(xa^{2k+m})^{-1}, a^{-1} \right) \quad . \quad (10)$$

Here k is *arbitrary* integer, $m = 0$ (*direct* product) or $m = 1$ (*wreath* product).

4 Discrete Gauge Principle

In fact, the gauge principle expresses the very general idea that any observable data can be presented in different “frames” at different points of space and

time, and there should be some way to compare these data. At the set-theoretic level, i.e., in the form suitable for both discrete and continuous cases, the main concepts of the gauge principle can be reduced to the following elements

- a set X , space or space-time;
- a set Σ , local states;
- the set Σ^X of Σ -valued functions on X , the set of states of *dynamical system*;
- a group $W \leq \text{Sym}(\Sigma^X)$ acting on Σ^X , *symmetries of the system*;
- identification of data describing dynamical system with states from Σ^X makes sense only modulo symmetries from W ;
- having no *a priori* connection between data from Σ^X at different points x and y in time and space we impose this *connection* (or *parallel transport*) explicitly as W -valued functions on edges of *abstract graph*:

$$P(x, y) \in W, \quad \varsigma(y) = \sigma(x)P(x, y) \ ;$$

connection $P(x, y)$ has obvious property $P(y, x) = P(x, y)^{-1}$;

- connection $P(x, y)$ is called *trivial* if it can be expressed in terms of a function on *vertices* of the graph: $P(x, y) = p(x)^{-1}p(y)$, $p(x), p(y) \in W$;
- invariance with respect to gauge symmetries depending on time or space $u(x), u(y) \in W$ leads to transformation rule for connection

$$P(x, y) \rightarrow u(x)^{-1}P(x, y)u(y); \tag{11}$$

- the *curvature* of connection $P(x, y)$ is defined as the conjugacy class of the *holonomy* along a cycle of a graph:

$$P(x_1, x_2, \dots, x_k) = P(x_1, x_2)P(x_2, x_3) \cdots P(x_k, x_1)$$

(the conjugacy means $P'(x_1, \dots, x_k) \sim u^{-1}P(x_1, \dots, x_k)u$ for any $u \in W$);

the curvature of trivial connection is obviously trivial: $\tilde{P}(x_1, \dots, x_k) \equiv 1$;

- the gauge principle does not tell us anything about the evolution of the connection itself, so gauge invariant relation describing dynamics of connection (*gauge field*) should be added.

Let us give two illustrations of how these concepts work in continuous case.

Electrodynamics. Abelian prototype of all gauge theories. Here the set X is 4-dimensional Minkowski space with points $x = (x^\mu)$ and the set of states is Hilbert space of complex scalar (Schrödinger equation) or spinor (Dirac equation) fields $\psi(x)$. The symmetry group of the Lagrangians and physical observables is $W = \text{U}(1)$. The elements of $\text{U}(1)$ can be represented as $e^{-i\alpha}$.

Let us make these elements dependent on space-time and consider the parallel transport for two closely situated space-time points:

$$P(x, x + \Delta x) = e^{-i\rho(x, x + \Delta x)} \ .$$

Specializing transformation rule (11) to this particular case

$$P'(x, x + \Delta x) = e^{i\alpha(x)} P(x, x + \Delta x) e^{-i\alpha(x+\Delta x)} ,$$

substituting approximations

$$\begin{aligned} P(x, x + \Delta x) &= e^{-i\rho(x, x+\Delta x)} \approx 1 - iA(x)\Delta x , \\ P'(x, x + \Delta x) &= e^{-i\rho(x, x+\Delta x)} \approx 1 - iA'(x)\Delta x , \\ e^{-i\alpha(x+\Delta x)} &\approx e^{-i\alpha(x)} (1 - i\nabla\alpha(x)\Delta x) , \end{aligned}$$

and taking into account commutativity of $W = U(1)$ we obtain

$$A'(x) = A(x) + \nabla\alpha(x) \quad \text{or, in components,} \quad A'_\mu(x) = A_\mu(x) + \frac{\partial\alpha(x)}{\partial x^\mu} . \quad (12)$$

The 1-form A taking values in the Lie algebra of $U(1)$ and its differential $F = (F_{\mu\nu}) = dA$ are identified with the electromagnetic *vector potential* and *field strength*, respectively. To provide the gauge invariance of the equations for field $\psi(x)$ we should replace partial by covariant derivatives

$$\partial_\mu \rightarrow D_\mu = \partial_\mu - iA_\mu(x)$$

in those equations.

Finally, evolution equations for the gauge field $A(x)$ should be added. In the case of electromagnetics these are Maxwell's equations:

$$dF = 0 \quad \text{first pair} \quad (13)$$

$$d \star F = 0 \quad \text{second pair.} \quad (14)$$

Here \star is the *Hodge conjugation* (*Hodge star operator*). Note that equation (14) corresponds to *vacuum Maxwell's equations*. In the presence of the *current* J the *second pair* takes the form $\star d \star F = J$. Note also that the *first pair* is essentially *a priori* statement, it reflects simply the fact that F , by definition, is the differential of an exterior form.

Non-Abelian gauge theories in continuous space-time. Only minor modifications are needed for the case of non-Abelian Lie group W . Again expansion of the W -valued parallel transport for two close space-time points x and $x + \Delta x$ with taking into account that $P(x, x) = 1$ leads to introducing of a Lie algebra valued 1-form $A = (A_\mu)$:

$$P(x, x + \Delta x) \approx 1 + A_\mu(x)\Delta x^\mu .$$

Infinitesimal manipulations with formula (11)

$$u(x)^{-1} P(x, x + \Delta x) u(x + \Delta x) \longrightarrow u(x)^{-1} (1 + A_\mu(x)\Delta x^\mu) \left(u(x) + \frac{\partial u(x)}{\partial x^\mu} \Delta x^\mu \right)$$

lead to the following transformation rule

$$A'_\mu(x) = u(x)^{-1}A_\mu(x)u(x) + u(x)^{-1}\frac{\partial u(x)}{\partial x^\mu} . \tag{15}$$

The curvature 2-form

$$F = dA + [A \wedge A]$$

is interpreted as *physical strength field*. In particular, the *trivial* connection

$$\tilde{A}_\mu(x) = u_0(x)^{-1}\frac{\partial u_0(x)}{\partial x^\mu}$$

is *flat*, i.e., its curvature $F = 0$.

There are different approaches to construct dynamical equations for gauge fields [2]. The most important example is *Yang-Mills theory* based on the Lagrangian

$$L_{YM} = \text{Tr} [F \wedge \star F] .$$

The Yang-Mills equations of motion read

$$dF + [A \wedge F] = 0 , \tag{16}$$

$$d \star F + [A \wedge \star F] = 0 . \tag{17}$$

Here again equation (16) is a *a priori* statement called *Bianchi identity*. Note that Maxwell's equations are a special case of Yang-Mills equations.

It is instructive to see what the Yang-Mills Lagrangian looks like in the discrete approximation. Replacing the Minkowski space X by a hypercubic lattice one can see that the discrete version of L_{YM} is proportional to $\sum_f \sigma(\gamma_f)$, where the summation is over all faces of a hypercubic constituent of the lattice;

$$\sigma = 2 \dim U - (\chi_U + \chi_{U^\dagger}) ;$$

χ_U and χ_{U^\dagger} are characters of the fundamental representation U of the gauge group and its dual representation, respectively; γ_f is the gauge group holonomy around the face f .

The Yang-Mills theory uses Hodge operation converting k -forms to $(n - k)$ -forms in n -dimensional space *with metric* $g_{\mu\nu}$. In topological applications so-called *BF theory* plays an important role since it does not involve a metric. In this theory, an additional dynamical field B is introduced. The Lie algebra valued $(n - 2)$ -form B and the 2-form F are combined into the Lagrangian $L_{BF} = \text{Tr} [B \wedge F] .$

5 Quantization Based on Finite Group

Quantization is a procedure for recovering a more fundamental quantum theory from its classical approximation. Both Lagrangian and Hamiltonian formulations of classical mechanics are based on the *principle of least action* which looks a

bit mysterious: a particle moving from one point to another “knows” in advance where it is going to arrive. Feynman’s path integral quantization [4] eliminates this apparent teleology in a quite natural way: classical trajectories correspond to the dominating (in the path integral) part of all possible trajectories.

Of course, recovering a theory from its approximation can not be performed uniquely. Moreover, discrepancies between a theory and its approximation may be essential. To illustrate, let us compare a simple discrete process with its approximation by continuous physical law.

5.1 Heat Equation from Bernoulli Trials

Let us consider a sequence of Bernoulli trials. The probability of a separate sequence is described by the *binomial distribution*

$$P(n_-, n_+) = \frac{(n_- + n_+)!}{n_-! n_+!} p_-^{n_-} p_+^{n_+} . \tag{18}$$

Here $\{-, +\}$ are possible outcomes of a single trial; p_-, p_+ are probabilities ($p_- + p_+ = 1$) and n_-, n_+ are numbers of the outcomes.

Applying Stirling’s approximation to (18) and introducing new variables $x = n_+ - n_-, t = n_- + n_+, v = p_- - p_+$ — let us call them “space”, “time” and “velocity”, respectively — we obtain

$$P(x, t) \approx \tilde{P}(x, t) = \frac{1}{\sqrt{1-v^2}} \sqrt{\frac{2}{\pi t}} \exp \left\{ -\frac{1}{2t} \left(\frac{x-vt}{\sqrt{1-v^2}} \right)^2 \right\} . \tag{19}$$

This is the *fundamental solution* of the *heat* (also known as *diffusion* or *Fokker-Planck*) *equation*:

$$\frac{\partial \tilde{P}(x, t)}{\partial t} + v \frac{\partial \tilde{P}(x, t)}{\partial x} = \frac{(1-v^2)}{2} \frac{\partial^2 \tilde{P}(x, t)}{\partial x^2} . \tag{20}$$

Note that expression (19) contains “relativistic” fragment $\frac{x-vt}{\sqrt{1-v^2}}$ due to the velocity limits $-1 \leq v \leq 1$ in our model. Note also that at $|v| = 1$ equation (20) reduces to the *wave equation*

$$\frac{\partial \tilde{P}(x, t)}{\partial t} \pm \frac{\partial \tilde{P}(x, t)}{\partial x} = 0 . \tag{21}$$

Now let us set a problem as is typical in mechanics: find extremal trajectories connecting two fixed points $(0, 0)$ and (X, T) . We adopt here the search of trajectories with maximum probability as a version of the “least action principle”. The probability of trajectory passing through some intermediate point (x, t) is the following *conditional probability*

$$\begin{aligned} P_{(0,0) \rightarrow (x,t) \rightarrow (X,T)} &= \frac{P(x, t) P(X-x, T-t)}{P(X, T)} \\ &= \frac{t!(T-t)! \left(\frac{T-X}{2}\right)! \left(\frac{T+X}{2}\right)!}{\left(\frac{t-x}{2}\right)! \left(\frac{t+x}{2}\right)! \left(\frac{T-t}{2} - \frac{X-x}{2}\right)! \left(\frac{T-t}{2} + \frac{X-x}{2}\right)! T!} . \tag{22} \end{aligned}$$

The conditional probability computed for approximation (19) takes the form

$$\tilde{P}_{(0,0) \rightarrow (x,t) \rightarrow (X,T)} = \frac{T}{\sqrt{\frac{\pi}{2}(1-v^2)tT(T-t)}} \exp \left\{ -\frac{(Xt - xT)^2}{2(1-v^2)tT(T-t)} \right\}. \quad (23)$$

One can see essential differences between (22) and (23):

- exact probabilities (22) do not depend on the velocity v (or on the probabilities p_-, p_+ of a single trial), whereas (23) contains artificial dependence,
- it is easy to check that expression (22) allows many trajectories with the same maximum probability, whereas extremals of (23) are deterministic trajectories, namely, straight lines $x = \frac{X}{T}t$.

These artifacts show that an important guiding principle of quantization — correspondence with classical limit — may not be quite reliable.

5.2 Gauge Connection and Quantization

The Aharonov–Bohm effect (Fig. 1) is one of the most striking illustrations of interplay between quantum behavior and gauge connection. Charged particles moving through the region containing perfectly shielded thin solenoid produce different interference patterns on a screen depending on whether the solenoid is turned on or off. There is no electromagnetic force acting on the particles, but working solenoid produces U(1)-connection adding or subtracting phases of the particles and, thus, changing the interference pattern.

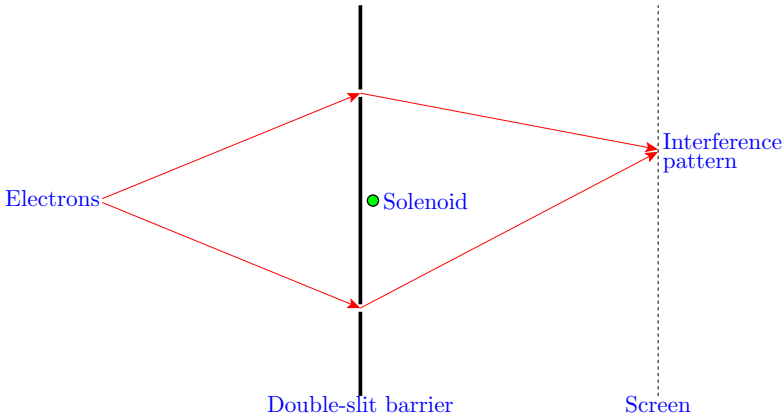


Fig. 1. Aharonov–Bohm effect. Magnetic flux is confined within the perfectly shielded solenoid; interference pattern is shifted in spite of absence of electromagnetic forces acting on the particles.

In the discrete time Feynman’s path amplitude decomposes into the product of elements of the group U(1) (or, more precisely, elements of the fundamental representation of U(1)):

$$\Phi = \exp(iS) = \exp\left(i \int L dt\right) \longrightarrow e^{iL_{0,1}} \dots e^{iL_{t-1,t}} \dots e^{iL_{T-1,T}} . \quad (24)$$

By the notation $L_{t-1,t}$ we emphasize that the Lagrangian is in fact a function defined on pairs of points (graph edges) — this is compatible with physics where the typical Lagrangians are determined by the *first order* derivatives. Thus, the expression $P(t-1, t) = e^{iL_{t-1,t}} \in U(1)$ can be interpreted as $U(1)$ -parallel transport.

We can introduce quantum mechanical description of a discrete system interpreting states $\sigma \in \Sigma$ as basis elements of a Hilbert space Ψ . This allows to describe statistics of observations of σ 's in terms of the *inner product* in Ψ .

Now let us replace expression (24) for Feynman's path amplitude by the following parallel transport along the path

$$\Phi = \rho(\alpha_{0,1}) \dots \rho(\alpha_{t-1,t}) \dots \rho(\alpha_{T-1,T}) .$$

Here $\alpha_{t-1,t}$ are elements of a *finite group* Γ — we shall call Γ *quantizing group* — and ρ is an unitary representation of Γ on the space Ψ .

Let us recall main properties of linear representations of finite groups [5].

- First of all, any linear representation of finite group is equivalent to unitary.
- Any unitary representation ρ is determined uniquely (up to isomorphism) by its *character* defined as $\chi_\rho(\alpha) = \text{Tr}\rho(\alpha)$, $\alpha \in \Gamma$.
- All values of χ_ρ and eigenvalues of ρ are elements of the ring \mathbb{A} of *algebraic integers*, moreover the eigenvalues are *roots of unity*. Recall that the ring \mathbb{A} consists of the roots of *monic* polynomials with integer coefficients [3].
- If all different irreducible representations of Γ are $\rho_1, \dots, \rho_i, \dots, \rho_h$ and $d_i = \dim \rho_i$, $M = |\Gamma|$ then

$$\sum_{i=1}^h d_i^2 = M \text{ and any } d_i \text{ divides } M : d_i \mid M.$$

- Any function $\varphi(\alpha)$ depending only on conjugacy classes of Γ , i.e., $\varphi(\beta^{-1}\alpha\beta) = \varphi(\alpha)$, is linear combination of characters $\chi_{\rho_1}, \dots, \chi_{\rho_h}$. Such functions are called *central* or *class* functions.

If the group Γ consists of M elements $\gamma_0, \dots, \gamma_{M-1}$ and n_k is the number of paths with the “phase” $\Phi = \rho(\gamma_k)$ at the point of observation (x, t) , then the amplitude at this point is $A = \sum_{k=0}^{M-1} n_k \rho(\gamma_k) \psi$, where $\psi \in \Psi$. The square of the amplitude (i.e., probability after appropriate normalization) can be written as

$$\langle A\psi | A\psi \rangle = \sum_{k=0}^{M-1} n_k^2 |\psi|^2 + \sum_{\substack{\gamma_i, \gamma_k \in \Gamma \\ i < k}} n_i n_k \langle \psi | \rho(\gamma_i^{-1} \gamma_k) + \rho^\dagger(\gamma_i^{-1} \gamma_k) | \psi \rangle , \quad (25)$$

or, after collecting like terms, as

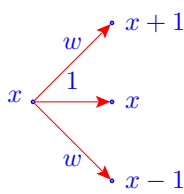
$$\langle A\psi|A\psi\rangle = \sum_{k=0}^{M-1} N_k(n_0, \dots, n_{M-1}) \langle \psi | \rho(\gamma_k) + \rho^\dagger(\gamma_k) | \psi \rangle, \quad (26)$$

where $N_k(n_0, \dots, n_{M-1})$ are quadratic polynomials with integer coefficients and arguments. Thus, algebraic integers are sufficient for all our computations except for normalization of probabilities requiring the quotient field of the ring \mathbb{A} .

5.3 Simple Model Inspired by Free Particle

In quantum mechanics – as is clear from the *never vanishing* expression $\exp\left(\frac{i}{\hbar}S\right)$ for the path amplitude – transitions from one to any other state are possible in principle. However, we shall consider computationally more tractable models with restricted sets of possible transitions.

Let us consider quantization of a free particle moving in one dimension. Such a particle is described by the Lagrangian $L = \frac{m\dot{x}^2}{2}$. Keeping only transitions to the closest points in the discretized space we come to the following rule for the one-time-step transition amplitudes



$e^{\frac{i}{\hbar} \frac{m\{(x+1)-x\}^2}{2}} = e^{i\frac{m}{2\hbar}}$
$e^{\frac{i}{\hbar} \frac{m\{x-x\}^2}{2}} = 1$
$e^{\frac{i}{\hbar} \frac{m\{(x-1)-x\}^2}{2}} = e^{i\frac{m}{2\hbar}}$

That is, we have evolution rule as an $U(1)$ -valued function R defined on pairs of points (graph edges). Symbolically:

$$R(x \rightarrow x) = 1 \in U(1),$$

$$R(x \rightarrow x - 1) = R(x \rightarrow x + 1) = w = e^{i\frac{m}{2\hbar}} \in U(1). \quad (27)$$

Now let us assume that w in (27) is an element of some representation of a finite group: $w = \rho(\alpha)$, $\alpha \in \Gamma = \{\gamma_0 = 1, \dots, \gamma_{M-1}\}$. Rearranging *multinomial coefficients* — *trinomial* in this concrete case — it is not difficult to write the sum amplitude over all paths of the form $(0, 0) \rightarrow (x, t)$

$$A_x^t(w) = \sum_{\tau=0}^t \frac{\tau!}{\left(\frac{\tau-x}{2}\right)! \left(\frac{\tau+x}{2}\right)!} \times \frac{t!}{\tau!(t-\tau)!} w^\tau. \quad (28)$$

Note that x must lie in the limits determined by t : $x \in [-t, t]$.

One of the most expressive peculiarities of quantum-mechanical behavior is the *destructive interference* — cancellation of non-zero amplitudes attached to different paths converging to the same point. By construction, the sum of amplitudes in our model is a function $A(w)$ depending on distribution of sources

of the particles, their initial phases, gauge fields acting along the paths, restrictions – like, e.g., “slits” – imposed on possible paths, etc. In the case of one-dimensional representation, the function $A(w)$ is a polynomial with algebraic integer coefficients, and w is a root of unity. Thus, the condition for destructive interference can be expressed by the system of polynomial equations: $A(w) = 0$ and $w^M = 1$. For concreteness let us consider the cyclic group $\Gamma = \mathbb{Z}_M = \{\gamma_0, \dots, \gamma_k, \dots, \gamma_{M-1}\}$. Any of its M irreducible representations takes the form $\rho(\gamma_k) = w^k$, where w is one of the M th roots of unity. For simplicity let w be the *primitive root*: $w = e^{2\pi i/M}$. Fig. 2 shows all possible transitions from the point x in three time steps with their amplitudes.

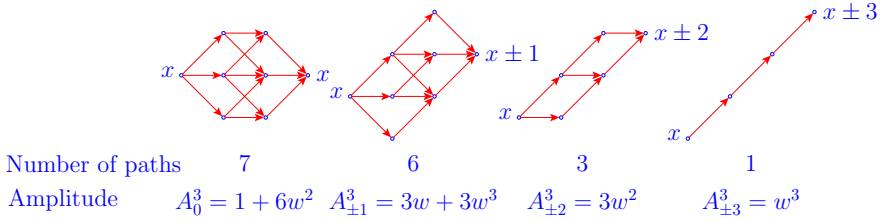


Fig. 2. Amplitudes for all possible paths in three time steps

We see that here only the conditions $A_{\pm 1}^3 = 3w + 3w^3 = 0$ and $w^M = 1$ with $M = 4$ can provide the destructive interference. Thus, for the model under consideration the natural quantizing group is \mathbb{Z}_4 . It is difficult to analyse more complicated combinations of paths in general, but computational experiments with the help of our C program support the assumption that the group \mathbb{Z}_4 is the only cyclic group providing quantum-mechanical behavior in this model.

Fig. 3 shows interference patterns — normalized squared amplitudes (“probabilities”) — from two sources placed in the positions $x = -4$ and $x = 4$ for 20 time steps. The upper and lower graph show interference pattern when sources are in the same ($\Delta\phi = 0$) and in the opposite ($\Delta\phi = \pi$) phases, respectively.

5.4 Generalization: Local Quantum Model on Regular Graph

The above model — with quantum transitions allowed only within the neighborhood of a vertex of a 1-dimensional lattice — can easily be generalized to arbitrary regular graph. Our definition of *local quantum model on k -valent graph* includes the following:

1. *Space* $X = \{x_1, \dots, x_N\}$ is a k -valent graph.
2. *Set of local transitions* $E_i = \{e_{0,i}, e_{1,i}, \dots, e_{k,i}\}$ is the set of k adjacent to the vertex x_i edges $e_{m,i} = (x_i \rightarrow x_{m,i})$ completed by the edge $e_{0,i} = (x_i \rightarrow x_i)$.
3. We assume that the *space symmetry* group $G = \text{Aut}(X)$ acts transitively on the set $\{E_1, \dots, E_N\}$.
4. $G_i = \text{Stab}_G(x_i) \leq G$ is the *stabilizer* of x_i ($g \in G_i$ means $x_i g = x_i$).
5. $\Omega_i = \{\omega_{0,i}, \omega_{1,i}, \dots, \omega_{h,i}\}$ is the *set of orbits* of G_i on E_i .

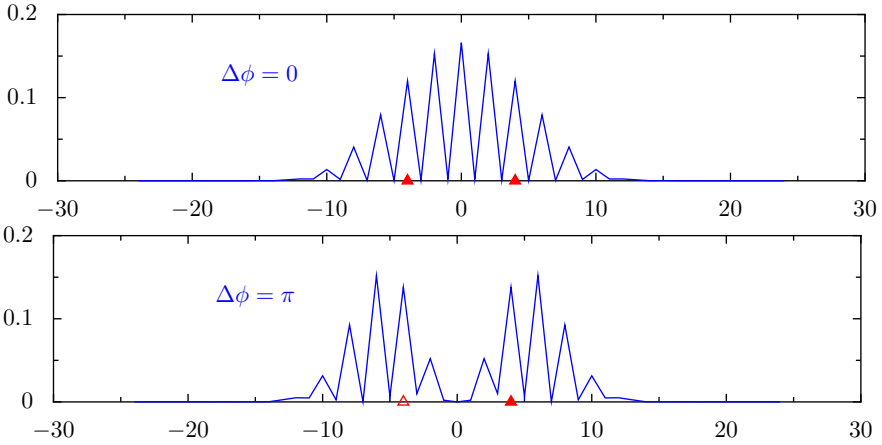
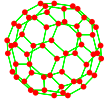


Fig. 3. Group \mathbb{Z}_4 . Interference from two sources. Number of time steps $T = 20$. Source positions are -4 and 4. Phase differences $\Delta\phi = \phi_4 - \phi_{-4}$ between sources are 0 and π .

- 6. *Quantizing group* Γ is a finite group: $\Gamma = \{\gamma_0, \dots, \gamma_{M-1}\}$.
- 7. *Evolution rule* R is a function on E_i with values in some representation $\rho(\Gamma)$. The rule R prescribes $\rho(\Gamma)$ -weights to the one-time-step transitions from x_i to elements of the neighborhood of x_i . From the symmetry considerations, R must be a function on orbits from Ω_i , i.e., $R(e_{m,i}g) = R(e_{m,i})$ for $g \in G_i$.

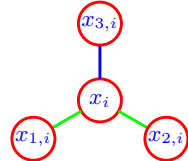
To illustrate these constructions, let us consider the local quantum model on the graph of *buckyball*. The incarnations of this 3-valent graph include, in particular:

- the *Caley graph of the icosahedral group* $\text{Alt}(5)$ (in mathematics);
- the *molecule* C_{60} (in carbon chemistry).

Here the space $X = \{x_1, \dots, x_{60}\}$ has the shape  and its symmetry

group is $G = \text{Aut}(X) = \mathbb{Z}_2 \times \text{Alt}(5)$. The set of local transitions takes the form $E_i = \{e_{0,i}, e_{1,i}, e_{2,i}, e_{3,i}\}$, where $e_{0,i} = (x_i \rightarrow x_i)$, $e_{1,i} = (x_i \rightarrow x_{1,i})$,

$e_{2,i} = (x_i \rightarrow x_{2,i})$, $e_{3,i} = (x_i \rightarrow x_{3,i})$ in accordance with



The stabilizer of x_i is $G_i = \text{Stab}_G(x_i) = \mathbb{Z}_2$. The set of orbits of G_i on E_i contains 3 orbits: $\Omega_i = \{\omega_{0,i} = \{e_{0,i}\}, \omega_{1,i} = \{e_{1,i}, e_{2,i}\}, \omega_{2,i} = \{e_{3,i}\}\}$, i.e., the stabilizer does not move the edges $(x_i \rightarrow x_i)$ and $(x_i \rightarrow x_{3,i})$ and swaps $(x_i \rightarrow x_{1,i})$ and $(x_i \rightarrow x_{2,i})$. This asymmetry results from different roles the edges play in the structure of the buckyball: $(x_i \rightarrow x_{1,i})$ and $(x_i \rightarrow x_{2,i})$ are edges of a pentagon adjacent to x_i , whereas $(x_i \rightarrow x_{3,i})$ separates two hexagons; in the carbon molecule C_{60} the edge $(x_i \rightarrow x_{3,i})$ corresponds to the double bond, whereas others are the single bonds.

The evolution rule takes the form:

$$\begin{aligned} R(x_i \rightarrow x_i) &= \rho(\alpha_0), \\ R(x_i \rightarrow x_{1,i}) &= R(x_i \rightarrow x_{2,i}) = \rho(\alpha_1), \\ R(x_i \rightarrow x_{3,i}) &= \rho(\alpha_2), \end{aligned}$$

where $\alpha_0, \alpha_1, \alpha_2 \in \Gamma$. If we take a one-dimensional representation and move α_0 – using gauge invariance – to the identity element of Γ , we see that the rule R depends on $v = \rho(\alpha_1)$ and $w = \rho(\alpha_2)$. Thus, the amplitudes in the quantum model on the buckyball take the form $A(v, w)$ depending on two roots of unity.

6 Conclusion

Extraordinary success of gauge theories in fundamental physics suggests that the gauge principle may be useful in theory and applications of discrete dynamical systems also. Furthermore, discrete and finite background allowing comprehensive study – especially with the help of computer algebra and methods of computational group theory – may lead to a deeper understanding of the gauge principle itself and its connection with the quantum behavior. To study more complicated models we are developing the C program.

Acknowledgments

The author thanks Laurent Bartholdi, Vladimir Gerdt, and Nikolai Vavilov for useful remarks and comments. This work was supported in part by the grants 07-01-00660 from the Russian Foundation for Basic Research and 1027.2008.2 from the Ministry of Education and Science of the Russian Federation.

References

1. O’Raifeartaigh, L., Straumann, N.: Gauge theory: Historical Origins and Some Modern Developments. *Reviews of Modern Physics* 72(1), 1–23 (2000)
2. Oeckl, R.: *Discrete Gauge Theory (From Lattices to TQPT)*. Imperial College Press, London (2005)
3. Kirillov, A.A.: *Elements of the Theory of Representations*. Springer, Berlin (1976)
4. Feynman, R.P., Hibbs, A.R.: *Quantum Mechanics and Path Integrals*. McGraw-Hill, New York (1965)
5. Serre, J.-P.: *Linear Representations of Finite Groups*. Springer, Heidelberg (1977)

Effective Quantifier Elimination for Presburger Arithmetic with Infinity

Aless Lasaruk¹ and Thomas Sturm²

¹ FORWISS, Universität Passau, 94030 Passau, Germany
lasaruk@uni-passau.de

² Departamento de Matemáticas, Estadística y Computación, Facultad de Ciencias,
Universidad de Cantabria, 39071 Santander, Spain
sturmt@unican.es

Abstract. We consider Presburger arithmetic extended by infinity. For this we give an effective quantifier elimination and decision procedure which implies also the completeness of our extension. The asymptotic worst-case complexity of our procedure is bounded by a function that is triply exponential in the input word length, which is known to be a tight bound for regular Presburger arithmetic. Possible application areas include quantifier elimination and decision procedures for Boolean algebras with cardinality constraints, which have recently moved into the focus of computer science research for software verification, and deductive database queries.

1 Introduction

The systematic investigation of the additive theory of integers with congruences started in 1929 with the pioneering work of Presburger [1]. The title of Presburger’s original work is an understatement. It only mentions the completeness of Presburger arithmetic. For his proof Presburger gave a decision procedure proving every sentence equivalent to either “true” or “false.” That is, the result is much more algorithmic than can be expected from complete theories in general. Even more important, from a modern point of view Presburger even gave a quantifier elimination procedure for a slightly extended language containing congruences. This on the one hand has considerable model theoretic consequences besides completeness, viz. substructure completeness, and on the other hand it considerably extends the power of a pure decision procedure with respect to possible practical applications.

For a long time already, Presburger arithmetic has been in the focus of mathematical as well as computer science research [2,3,4,5,6,7,8]. Quite recently the authors of the present note have extended quantifier elimination for Presburger arithmetic to parametric multiplicative constants [9] and furthermore to certain nonlinear input formulas [10].

Applications quite naturally arise in many areas of science and engineering. This might explain also the remarkable historical fact that already in 1954 Davis had implemented Presburger’s original algorithm on a digital computer [11]. In

computer science many array subscript calculations fall within the region of problems decidable by Presburger arithmetic. This observation plays a prominent role in several proof of correctness systems for computer programs, beginning with the Stanford Pascal Verifier [12] and recently including Microsoft’s Spec#. Further recent applications include elimination procedures for Boolean algebras with cardinality constraints in the context of deductive database queries [13].

Our present work extends classical Presburger arithmetic PA to PAI by adding a single infinite number with trivial arithmetic. This is inspired by the idea to include statements on the infinity of certain sets in deductive database queries [13]. Independently, Boolean algebras in combination with Presburger arithmetic have recently been considered for software verification. Within that framework there has even been considered our idea of including infinity. This was, however, realized in the Boolean algebra sort in contrast to the Presburger sort [14,15,16].

Some additional mathematical applications of our work might arise from the fact that in elimination procedures for valued fields, field quantifiers are often eliminated in favour of quantifiers over the value group [17]. Such procedures can possibly be simplified by not treating 0, which has value ∞ , specially. Notice that the value group of the important class of p -adic valuations is in fact the ordered additive group of the integers such that the set of possible values ranges over $\mathbb{Z} \cup \{\infty\}$.

The outline of our paper is as follows: In Section 2 we make precise our extension of Presburger arithmetic by infinity and its relation to Presburger’s original work. In Section 3 we introduce essential equivalence transformations on and normal forms of formulas in our extended framework, which are fundamental on the one hand for our elimination procedure and on the other hand for any serious attempt of an implementation. Section 4 states our main result, which is an effective quantifier elimination procedure for PAI and the model theoretic consequences of this. Section 5 gives asymptotic upper bounds on the worst-case complexity of our procedure. These bounds are known to be tight for PA. Section 6 give some examples for quantifier elimination in PAI in order to illustrate the procedure, to give an intuition of the semantics, and to point at possible applications. In Section 7 we finally summarize our results and indicate future research directions.

2 Presburger Arithmetic with Infinity

From a precise model-theoretic point of view, Presburger arithmetic is the model class of the axioms originally given by Presburger [1]. Presburger had considered congruences $a \equiv_{\alpha} b$ as abbreviations for first-order sentences $\exists x(\alpha \odot x + a = b)$, which he—besides atomic formulas—referred to as “ground formulas.” This allowed him on the one hand to use the finite and apparently more natural language $(0, 1, +, \leq)$ of additive ordered groups and on the other hand not to emphasize too much the role of congruences in his procedures as they were considered only intermediate objects for his *decision procedure*. It is noteworthy that Presburger’s presentation does not explicitly treat the ordering “ \leq ” at all.

There is only a remark in the appendix that the procedure can be generalized accordingly. This generalization has been made explicit e.g. in a textbook by Monk [18].

From a modern quantifier elimination point of view it is exactly the formal manifestation of congruences in an extended language

$$L = (0, 1, +, -, \leq, \equiv)$$

that straightforwardly exhibits that Presburger’s decision procedure is actually even a quantifier elimination procedure over L . Notice that for convenience we have also added a function “ $-$ ” for the additive inverse. We are now going to list Presburger’s set of axioms with some slight modifications: We drop six axioms referring to the semantics of logical connectives and equality as their semantics is usually fixed on the meta-mathematical level in modern algebraic model theory. We use congruences in contrast to their existentially quantified equivalents described above. We add one axiom for our additional function symbol “ $-$.” Concerning notation, we use the symbol “ \odot ” to emphasize that any “multiplication” $\alpha \odot a$ occurring here is in fact an abbreviated notation for a corresponding addition $a + \dots + a$ (α times). Furthermore, we use modern logical symbols and infix notation, and we save parentheses by following common conventions that the precedence of our logical operators decreases in the order “ \leq ,” “ $=$,” “ \neg ,” “ \wedge ,” “ \vee ,” “ \longrightarrow ,” “ \longleftarrow .”

- (π_1) $a + c = b + c \longrightarrow a = b$
- (π_2) $a + b = b + a$
- (π_3) $a + (b + c) = (a + b) + c$
- (π_4) $a + 0 = a$
- (π_5) $\exists b(a + b = c)$
- (π_6) $\alpha \odot a = \alpha \odot b \longrightarrow a = b$ for $\alpha = 2, 3, \dots$
- (π_7) $a \equiv_\alpha 0 \vee a \equiv_\alpha 1 \vee \dots \vee a \equiv_\alpha (\alpha - 1) \odot 1$ for $\alpha = 2, 3, \dots$
- (π_8) $\neg \alpha \odot 1 = 0$ for $\alpha = 2, 3, \dots$
- (π_9) $a + (-a) = 0$.

Note that in (π_6)–(π_8) we have countably infinite subsets of axioms and, consequently, the entire set of axioms is countably infinite. For reasons discussed above this system does not include any axioms for the ordering. Since for our work here it is not necessary to refer to explicit axioms at all, we pragmatically switch to the countably infinite set $\Pi = \{ \varphi \mid \varphi \text{ is } L\text{-formula and } \mathbb{Z} \models \varphi \}$, which obviously comprises π_1, \dots, π_9 . We define PA to denote the *model class* of Π :

$$\text{PA} = \text{Mod}(\Pi) = \{ \mathbb{A} \mid \mathbb{A} \text{ } L\text{-structure and } \mathbb{A} \models \Pi \}.$$

We obviously have $\mathbb{Z} \in \text{PA}$, and thus Π is consistent. The main result stated by Presburger was the fact that Π is even *complete*. That is, every L -sentence is either valid or invalid in all L -structures in PA simultaneously. In the former case, one writes $\text{PA} \models \varphi$ or—since PA is completely determined by Π —shortly $\Pi \models \varphi$. Since Π is first-order and recursively enumerable, it follows from this

completeness that PA is *decidable*. That is, there is an algorithm with input a first-order L -formula φ and output either \top or \perp , which always terminates and which returns \top if and only if $\text{PA} \models \varphi$.

We would like to remind the reader at this point that there is no simple correspondence between completeness and decidability in the sense defined above. For instance, the theory of algebraically closed fields is decidable but not complete while the theory axiomatized by all first-order $(0, s)$ -sentences valid in \mathbb{N} is complete but not decidable.

From a modern point of view, Presburger has even shown that PA is *substructure complete*. That is, for any two L -structures \mathbb{A} and \mathbb{B} in PA that have a common substructure one may add to L constants for all elements of this common substructure yielding L' . When then viewing \mathbb{A} and \mathbb{B} as L' -structures in a natural way even all L' -formulas will be either valid or invalid in both \mathbb{A} and \mathbb{B} simultaneously. Substructure completeness is equivalent to the existence of a quantifier elimination procedure, which Presburger has implicitly given for PA.

Since variable-free atomic formulas are decidable in PA, any quantifier elimination procedure yields a decision procedure via successive elimination of all variables. In fact, it does even more: Since every formula algorithmically turns out to be equivalent to either “true” or “false,” a return value \perp for input φ of such a decision procedure may be interpreted not only as the existence of $\mathbb{A}_0 \in \text{PA}$ with $\mathbb{A}_0 \not\models \varphi$ but it even follows that *for all* $\mathbb{A} \in \text{PA}$ we have $\mathbb{A} \not\models \varphi$. In other words, this exhibits also the completeness of PA.

To get a more precise picture of PA, recall the famous upward Löwenheim–Skolem Theorem [19,20,21]: From the existence of the infinite model $\mathbb{Z} \in \text{PA}$, it follows that PA contains models of arbitrary infinite cardinality. In particular, PA is a proper class in contrast to a set.

We are now going to modify and extend Presburger’s axioms Π to describe the integers with infinity in an extended language

$$L_\infty = (0, 1, \infty, +, -, \leq, \equiv).$$

We obtain (ι_1) by restricting the axioms in Π to finite numbers, and we add (ι_2) – (ι_8) to axiomatize ∞ . Let $V(\pi)$ denote the finite set of variables occurring freely in an L -formula π :

- $(\iota_1) \bigwedge_{v \in V(\pi)} \neg v = \infty \longrightarrow \pi$ for $\pi \in \Pi$
- $(\iota_2) a + \infty = \infty$
- $(\iota_3) \infty + a = \infty$
- $(\iota_4) -\infty = \infty$
- $(\iota_5) a \leq \infty$
- $(\iota_6) \neg a = \infty \longrightarrow \neg \infty \leq a$
- $(\iota_7) a \equiv_\alpha \infty$ for $\alpha = 2, 3, \dots$
- $(\iota_8) \infty \equiv_\alpha a$ for $\alpha = 2, 3, \dots$

We refer to this new countably infinite set of axioms as I and to its model class as PAI.

It is not hard to see that $\mathbb{Z} \cup \{\infty\}$ with trivial arithmetic on infinity as defined in (ι_2) – (ι_4) and relations as defined in (ι_5) – (ι_8) is in PAI and thus I is consistent: Since (ι_1) collects assertions about $\mathbb{Z} \cup \{\infty\}$ explicitly excluding ∞ the validity of these assertions for $\mathbb{Z} \cup \{\infty\}$ follows from the validity of Π for \mathbb{Z} . Axioms (ι_2) – (ι_4) straightforwardly complement the definitions of binary “+” and unary “−” at points involving infinity. Axioms (ι_5) and (ι_6) define the binary relation “ \leq ” at points involving infinity. They cannot contradict each other since the former refers to infinite right hand sides while the latter excludes this. Axioms (ι_7) and (ι_8) straightforwardly give a trivial definition of the congruence relation at points involving infinity. Observe that there are no axioms combining “+,” “−” involving infinity on the one hand with “ \leq ” or “ \equiv ” involving infinity on the other hand.

In analogy to PA our new model class PAI is a proper class containing models of arbitrarily large cardinality. In the next section we are going to devise a quantifier elimination procedure for PAI. This exhibits the substructure completeness of PAI. Since our axioms (ι_2) – (ι_8) obviously admit to decide variable-free atomic formulas involving infinity, it will follow that PAI is furthermore complete and decidable and our quantifier elimination procedure yields a corresponding decision procedure.

3 Normal Forms

As a preparation for our quantifier elimination procedure for PAI we are going to discuss in this section normal forms for L_∞ -formulas, the contained atomic formulas, and the terms in these atomic formulas.

3.1 Formulas

Our first goal is to isolate all occurrences of the L_∞ -constant ∞ . We call an L_∞ -formula in *normal form* if ∞ occurs there exclusively in equations $x = \infty$, where x is a variable. The following lemma guarantees that such normal forms generally exist:

Lemma 1 (Normalization of L_∞ -formulas). *Let φ be an atomic L_∞ -formula containing the L_∞ -constant ∞ . Then φ can be equivalently transformed into “true” or into an L -formula*

$$\bigvee_{x \in X} x = \infty,$$

where X is a subset of the variables occurring in φ .

Proof. To start with, observe that for L -terms u with variables x_1, \dots, x_n we have

$$\text{PAI} \models u = \infty \iff \bigvee_{i=1}^n x_i = \infty.$$

Our atomic L_∞ -formula φ is of the form $s \varrho t$, where s, t are L_∞ -terms and ϱ is one of $=, \leq, \equiv_\alpha$. From our observation above it follows that s or t can be equivalently replaced by ∞ when containing the L_∞ -constant ∞ . Next, congruences containing ∞ , atomic formulas $s \leq \infty$, and equations $\infty = \infty$ can be equivalently replaced by “true.” Atomic formulas $\infty \leq t$ can be equivalently replaced by $t = \infty$. So unless we have already evaluated to “true” we finally arrive at an equation of the form $u = \infty$ where u is an L -term, and we can once more apply our above observation. \square

It is interesting to observe that the use of the L_∞ -constant ∞ can in fact be avoided entirely: According to Lemma [1](#), we may assume w.l.o.g. that ∞ occurs exclusively in equations $x = \infty$, where x is a variable.

Lemma 2. *Let t be an L -term. Then $\text{PAI} \models t = \infty \iff \neg t - t = 0$.* \square

Furthermore, our language admits to bring quantifier-free L_∞ formulas into *positive normal form*, i.e., the only logical operators occurring are “ \wedge ” and “ \vee .” All Boolean connectives can be equivalently expressed by means of “ \wedge ,” “ \vee ,” and “ \neg .” Using de Morgan’s laws and involution, all “ \neg ” can be moved inside until they cancel or directly precede some atomic formula. Then we have:

Lemma 3 (Positive normal form). *Let s, t be L_∞ -terms. Then the following holds:*

- (i) $\text{PAI} \models \neg s \equiv_\alpha t \iff \bigvee_{\beta=1}^{\alpha-1} s \equiv_\alpha t + \beta \wedge s - s = 0 \wedge t - t = 0$
- (ii) $\text{PAI} \models \neg s \leq t \iff t + 1 \leq s \wedge t - t = 0$
- (iii) $\text{PAI} \models \neg s = t \iff (t + 1 \leq s \wedge t - t = 0) \vee (s + 1 \leq t \wedge s - s = 0)$. \square

In particular from Lemma [2](#) and Lemma [3](#)(iii) we obtain

$$t = \infty \iff \neg t - t = 0 \iff 1 \leq t - t.$$

Finally, notice that our Lemmas [1](#)–[3](#) are compatible in the following sense: for quantifier-free L_∞ -formulas one can obtain positive normal forms not containing the L_∞ -constant ∞ .

3.2 Terms

Recall from Lemma [1](#) that we can bring all L_∞ -formulas into a normal form where the L_∞ -constant ∞ occurs exclusively in equations of the form $x = \infty$ where x is a variable, and all other atomic formulas are L -formulas. So for the discussion of normal forms of our terms it is sufficient to consider L -terms.

Notice that even for L -terms common transformations fail. At the first place, “ $-$ ” does not generally yield an additive inverse in PAI since $\infty + (-\infty) = \infty + \infty = \infty$. Consequently, simplifications of terms like replacing $x + (-x)$ by 0 are not sound.

The following lemma lists some relevant axioms in Π which remain valid in PAI and mentions some further arithmetic rules valid for PAI, which we are going to use in the sequel. When referring to elements $\alpha \in \mathbb{Z}$ occurring in some term, we consider α an abbreviated notation for the corresponding term of the form $\pm(1 + \dots + 1)$ or 0.

Lemma 4 (Arithmetic in PAI). *Presburger's axioms (π_2) – (π_4) , (π_6) – (π_8) remain valid in PAI. This includes the laws of commutativity and associativity and the neutrality of 0 for addition and the involutivity of “–.” Furthermore the following arithmetic rules hold:*

- (i) PAI $\models \alpha \odot 0 = 0$ for $\alpha \in \mathbb{N}$
- (ii) PAI $\models -0 = 0$
- (iii) PAI $\models -(-a) = a$
- (iv) PAI $\models -\alpha \odot a = \alpha \odot (-a)$ for $\alpha \in \mathbb{N}$
- (v) PAI $\models \alpha \odot a + (-\beta \odot a) = (\alpha - \beta) \odot a$ for $\alpha, \beta \in \mathbb{N}, \alpha > \beta$
- (vi) PAI $\models \alpha \odot a + (-\alpha \odot a) = a - a$ for $\alpha \in \mathbb{N}$
- (vii) PAI $\models a + \alpha = b + \alpha \longrightarrow a = b$ for $\alpha \in \mathbb{Z}$
- (viii) PAI $\models \alpha \odot (a + b) = \alpha \odot a + \alpha \odot b$ for $\alpha \in \mathbb{Z}$. □

When using n -ary addition in the sequel, we tacitly assume that “+” is right-associative. Due to Lemma [A](#)(π_2) this assumption is only of syntactic relevance. Furthermore, we are going to use $s - t$ as a short notation for terms $s + (-t)$.

Lemma 5 (Normal form of L -terms). *Let t be an L -term with variables x_1, \dots, x_n . Then there exist $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n \in \mathbb{N}$ and $\alpha \in \mathbb{Z}$ such that*

$$\text{PAI} \models t = \alpha + \sum_{i=1}^n \alpha_i x_i - \sum_{i=1}^n \beta_i x_i.$$

Proof. Use Lemma [A](#)(iv,viii) to move all occurrences of “–” inside until all such occurrences are nested occurrences in front of constants or variables. Use Lemma [A](#)(iii) to reduce each such nested occurrence to at most one. Use Lemma [A](#)(ii) and Lemma [A](#)(π_4) to eliminate all occurrences of 0. Use Lemma [A](#)(π_3) to obtain a right-associative n -ary sum of variables and the constant 1 possibly preceded by “–.” Use Lemma [A](#)(π_2, π_3) to reorder this n -ary sum as required by our normal form. Finally use (ι_9) and Lemma [A](#)(π_2, π_3, iv) to rewrite the initial sequence $\pm 1 \pm \dots \pm 1$ in this n -ary sum as $\pm(1 + \dots + 1)$. □

Lemma 6 (Unique normal form of L -terms). *Consider an L -term t with variables x_1, \dots, x_n , which we assume to be ordered $x_1 \prec x_2 \prec \dots \prec x_n$. Then there is an enumeration $y_1, \dots, y_k, \dots, y_n$ of $\{x_1, \dots, x_n\}$ with $y_1 \prec \dots \prec y_k$ and $y_{k+1} \prec \dots \prec y_n$, there is $\alpha \in \mathbb{Z}$, and there are $\alpha_1, \dots, \alpha_k \in \mathbb{Z}$ such that*

$$\text{PAI} \models t = \alpha + \sum_{i=1}^k \alpha_i y_i + \sum_{i=k+1}^n y_i - \sum_{i=k+1}^n y_i.$$

Furthermore, there is only one such choice $y_1, \dots, y_k, \dots, y_n, \alpha, \alpha_1, \dots, \alpha_k$. □

Notice that the normal form in the previous lemma depends on the considered set $\{x_1, \dots, x_n\}$ of variables. A *unique normal form* that does not depend on x_1, \dots, x_n is obtained by deleting summands $\alpha_i y_i$ where $\alpha_i = 0$. In addition, we agree to delete α if $\alpha = 0$.

3.3 Atomic Formulas

Notice that even for the atomic L -formulas in our normal form according to Lemma 6 and even when subtracting terms “carefully” in the sense of the previous section, some familiar equivalence transformations are not valid in PAI. For instance, $x = x + y$ is not equivalent to $y = 0$ as the interpretation $x = \infty$ admits arbitrary interpretations of y in the former equation. Normalization of right-hand sides to 0 in atomic L -formulas can, however, be achieved when making on the syntactic level case distinctions similar to those in the proof of Lemma 6.

Lemma 7 (Normal form of atomic L -formulas). *Let s, t be L -terms. Denote by $V(s)$ and $V(t)$ the finite sets of variables occurring in s and t , respectively. Then*

- (i) $\text{PAI} \models s \equiv_{\alpha} t \iff s - t \equiv_{\alpha} 0$
- (ii) $\text{PAI} \models s \leq t \iff s - t \leq 0 \vee \bigvee_{x \in V(t)} x = \infty$
- (iii) $\text{PAI} \models s = t \iff s - t = 0 \vee \left(\bigvee_{x \in V(s)} x = \infty \wedge \bigvee_{x \in V(t)} x = \infty \right)$.

Proof. Fix an interpretation of all variables so that $s, t \in \mathbb{Z} \cup \{\infty\}$. Recall from the proof of Lemma 6 that $s, t = \infty$ if and only if at least one variable in s, t , resp., is interpreted as infinity.

- (i) If one of $s = \infty$ or $t = \infty$, then $s - t = \infty$ and both $s \equiv_{\alpha} t$ and $s - t \equiv_{\alpha} 0$ are true. Otherwise $s, t \in \mathbb{Z}$, where our transformation is known to be correct.
- (ii) If $t = \infty$, then both sides of our equivalence are true. Assume now that $t \neq \infty$. If $s = \infty$ then both sides of our equivalence are false. Otherwise $s, t \in \mathbb{Z}$, where subtraction on both sides of the atomic formula is known to be correct, and our big disjunction is false.
- (iii) If $s = t = \infty$, then both sides of our equivalence are true. If w.l.o.g. $s = \infty$ and $t \neq \infty$, then both sides of our equivalence are false. Otherwise $s, t \in \mathbb{Z}$, where subtraction on both sides of the equation is known to be correct, and our big disjunctions are both false. □

This normal form for atomic formulas is very convenient for practical purposes. Recall that the practical applicability of quantifier elimination by virtual substitution crucially depends on powerful methods for simplification of intermediate results, and these methods in turn are typically based on atomic formulas with right-hand sides normalized to 0 [22].

For Presburger quantifier elimination, one temporarily renormalizes to $sx = t$ where x is the variable currently considered for elimination. The following lemma describes a corresponding normal form for PAI:

Lemma 8 (Normal form of atomic L -formulas w.r.t. a variable). *Let $t \geq 0$ be an atomic L -formula in normal form according to Lemma 7, let t be in unique normal form, and let x be a variable occurring in t . That is*

$$t = \alpha + \sum_{i=1}^k \alpha_i y_i + \sum_{i=k+1}^n y_i - \sum_{i=k+1}^n y_i,$$

and $y_j = x$ for one and only one $j \in \{1, \dots, n\}$. Then

$$\text{PAI} \models t \varrho 0 \longleftrightarrow \eta,$$

where η is a quantifier-free L_∞ -formula, where x occurs in η exclusively in atomic L -formulas $s \varrho u$ with

$$s = \begin{cases} \alpha_j x & \text{for } j \leq k \\ x - x & \text{for } j > k \end{cases} \quad \text{and} \quad u = -\alpha + \sum_{\substack{i=1 \\ i \neq j}}^k -\alpha_i y_i + \sum_{\substack{i=k+1 \\ i \neq j}}^n y_i - \sum_{\substack{i=k+1 \\ i \neq j}}^n y_i.$$

Proof. To start with, it is easy to see that $\text{PAI} \models t \varrho 0 \longleftrightarrow s - u \varrho 0$. We are now going to distinguish cases on ϱ :

$$\text{PAI} \models s - u \equiv_\alpha 0 \longleftrightarrow s \equiv_\alpha u$$

$$\text{PAI} \models s - u \leq 0 \longleftrightarrow s \leq u \wedge x - x = 0 \wedge u - u = 0$$

$$\text{PAI} \models s - u = 0 \longleftrightarrow s = u \wedge (x - x = 0 \vee u - u = 0).$$

Recall that $\text{PAI} \models r - r = 0 \longleftrightarrow \neg r = \infty$. □

3.4 Relevant Combinations of Normal Forms

Consider a quantifier-free L_∞ -formula φ . To put φ into *general normal form*, we apply Lemma 1 to isolate all occurrences of the L_∞ -constant ∞ , then apply Lemma 2 to normalize all right hand sides of contained atomic L -formulas to 0, and finally bring the left hand side terms of the contained atomic L -formulas into unique normal form.

Recall from Section 3.1 that we can bring our φ into positive normal form. Since none of the transformations leading to the general normal form defined above introduces any Boolean connectives except “ \wedge ” and “ \vee ,” we can obtain *positive general normal forms* by computing positive normal forms and subsequently computing general normal forms.

An *elimination normal form* with respect to some variable x is obtained from a general normal form of φ by applying Lemma 8 to equivalently replace all contained atomic L -formulas containing x . Notice that this preserves positivity as well. Hence we can obtain a *positive elimination normal form* with respect to x from a positive general normal form.

4 Quantifier Elimination

We are going to reuse here an essential part of our existing quantifier elimination procedure for regular Presburger arithmetic [9]. At least for complexity considerations it is going to be essential that this is a *virtual substitution* method [23,24,25,26,27,9,10]. Let us recall some basic facts on virtual substitution: The key idea for the elimination of a quantifier $\exists x$ from $\exists x \varphi$ is to compute a finite *elimination set* E such that

$$\exists x \varphi \longleftrightarrow \bigvee_{t \in E} \varphi[t/x],$$

i.e., there are finitely many terms substituted for x into φ , where E is constructed such that for any choice of parameters the following holds: If there exists some satisfying choice for x at all, then at least one $t \in E$ evaluates to a satisfying choice for x . The notion *virtual* refers to the following generalization:

1. The elimination set E need not exclusively contain regular terms but possibly also some *pseudo-terms*. A typical example are pseudo-terms containing division with real quantifier elimination [24]. Since elimination sets can comprise both regular terms and pseudo-terms, one often refers to their elements as test *points*.
2. Instead of regular substitution one uses a *modified substitution*, which does not map terms to terms but more generally atomic formulas to quantifier-free formulas. This happens in such a way that the substitution result, in contrast to the substituted test point t , generally does not contain any symbols not in the considered language.
3. The generalized test points are paired with *guards*, which ascertain their validity. For instance, for a test point containing division we would add the guard that the denominator is not zero. The guards are added conjunctively when substituting.

Altogether our substitution idea given above generalizes as follows:

$$\exists x\varphi \longleftrightarrow \bigvee_{(\gamma,t) \in E} \gamma \wedge \varphi[t//x].$$

Our elimination procedure is going to make use of our main technical Lemma for uniform Presburger arithmetic [9, Lemma 8]. This lemma uses a more general form of virtual substitution introducing bounded quantifiers. For the special case of classical Presburger arithmetic it can be adapted to the classical virtual substitution framework discussed above [9, Lemma 2(ii)]. We explicitly formulate this adapted result:

Lemma 9 (Regular Presburger elimination set). *Consider an L -formula $\exists x\varphi$ where φ is quantifier-free and in positive normal form. Let the set of all atomic formulas of φ that contain x be*

$$A = \{ \alpha_i x \varrho_i r_i \mid i \in I_1 \dot{\cup} I_2 \}.$$

We have $\alpha_i \in \mathbb{Z} \setminus \{0\}$, and the r_i do not contain the variable x . For $i \in I_1$, we have $\varrho_i \in \{=, \leq\}$. For $i \in I_2$, we have that ϱ_i is a congruence \equiv_{m_i} . Define $m = \text{lcm}\{ |m_i| : i \in I_2 \}$ where $\text{lcm} \emptyset := 1$. Then

$$E = \bigcup_{i \in I_1} \bigcup_{-|\alpha_i|m \leq k \leq |\alpha_i|m} \{(\gamma_i, t_i)\} \cup \bigcup_{0 \leq k < m} \{(\text{true}, k)\},$$

where $\gamma_i = (r_i + k \equiv_{|\alpha_i|} 0)$ and $t_i = \frac{r_i + k}{\alpha_i}$, is an elimination set for $\exists x\varphi$. \square

For our purposes here it is important to know that the elimination set E is computed essentially from the set of atomic formulas contained in φ in such a way that the following holds:

Remark 10. Fix an interpretation v of all variables into \mathbb{Z} for all variables, and consider the subset $A^+ = \{\psi \in A \mid (\text{PA}, v) \models \psi\}$ of all atomic formulas that hold with respect to this interpretation. Then there is $(\gamma, t) \in E$ such that the following holds:

- (i) $(\text{PA}, v) \models \gamma$ and $(\text{PA}, v) \models \psi[t//x]$ for all $\psi \in A^+$.
- (ii) All variables in (γ, t) occur also in A^+ . □

So with respect to any interpretation of variables where $x = z$, the relevant test term satisfies *at least* those atomic formulas that are satisfied by z —possibly more. This is the reason behind working with positive formulas when devising elimination sets.

Weispfenning originally had used *Skolem sets*, which exactly simulated the truth values of all atomic formulas [23]. He switched to positive formulas in the subsequent work on the reals [24]. In the context of valued fields, the second author introduced *CS-sets*, which further generalize the positive formula approach used here [26]. There has been also some research on taking into account the Boolean structure of φ for computing smaller elimination sets [24,28].

Remark 10(ii) is a quite natural property, which in fact holds for elimination sets for numerous theories. Nevertheless, it is not generally true but closely related to the question whether or not there are several atomic formulas combined to obtain some test point. This happened for example in the first elimination sets for the reals, where there were arithmetic means of interval boundaries computed in order to hit open intervals [23]. In elimination sets for discretely valued fields there are even up to three atomic formulas combined [26].

Lemma 11. *Consider an atomic L-formula $s \varrho t$ in normal form with respect to x :*

$$s \in \{\beta x, x - x\} \quad \text{and} \quad t = \alpha + \sum_{i=1}^k \alpha_i y_i + \sum_{i=k+1}^n y_i - \sum_{i=k+1}^n y_i.$$

Fix an interpretation $v : \{y_1, \dots, y_n\} \rightarrow \mathbb{Z} \cup \{\infty\}$. If $\infty \in v\{\{y_1, \dots, y_n\}\}$, then one and only one of the following two assertions is true:

$$(\text{PAI}, v \cup \{x = z\}) \models s \varrho t \quad \text{iff} \quad z = \infty, \quad (\text{PAI}, v) \models s \varrho t.$$

That is, with respect to v either $x = \infty$ is the only satisfying choice for $s \varrho t$ or any choice from $\mathbb{Z} \cup \{\infty\}$ will do.

Proof. With respect to v we obtain $s = \infty$, $s \leq \infty$, or $s \equiv_\alpha \infty$ depending on ϱ . In the first case, $x = \infty$ is the only solution. In the other cases any choice for x is valid. □

Lemma 12 (Elimination of one existential quantifier). *Let φ be a quantifier-free L_∞ -formula in positive elimination normal form with respect to x . Compute the set A of all those atomic L -formulas containing x , where the left hand side is not $x - x$. Compute an elimination set E for A according to Lemma 9. Then $E' := E \cup \{(\text{true}, 0), (\text{true}, \infty)\}$ is an elimination set for $\exists x\varphi$.*

Proof. Fix an interpretation into $\mathbb{Z} \cup \{\infty\}$ for all parameters, i.e. all variables except x . Assume that there is a satisfying interpretation $z \in \mathbb{Z} \cup \{\infty\}$ for x . Since φ is positive it suffices to substitute some test point for x such that at least those atomic formulas A^+ become true that contain x and hold for the choice $x = z$. Notice that in general A^+ is neither a subset nor a superset of A . If $z = \infty$, then $(\text{true}, \infty) \in E'$ is a suitable choice.

Assume now that $z \neq \infty$ and note that then $x = \infty \notin A^+$. Let A' be the subset of all atomic formulas in A not containing any parameter that is interpreted as ∞ . If $A' \cap A^+ \neq \emptyset$, then according to Remark 10(i) the regular Presburger elimination set E provides a test point t rendering true all atomic formulas in $A' \cap A^+$. Using Remark 10(ii) it follows that $t \in \mathbb{Z}$ with respect to our fixed interpretation. If, in contrast, $A' \cap A^+ = \emptyset$, then we may consider $t = 0$ as we have explicitly added $(\text{true}, 0)$ to E' . In either case for all atomic L -formulas in $A^+ \setminus A$, i.e., where the left hand side is $x - x$, the satisfying value z is exactly simulated by t yielding 0 in both cases. By Lemma 11 all atomic formulas in $(A \setminus A') \cap A^+$ are satisfied by any choice for x since they are satisfied by $z \neq \infty$. □

Theorem 13. *PAI admits effective quantifier elimination.*

Proof. Let $\hat{\varphi} = Q_1x_1 \dots Q_nx_n\varphi$ be an L_∞ -formula, which is w.l.o.g. in prenex normal form, i.e., φ is quantifier-free. We proceed by induction on the number n of quantifiers in $\hat{\varphi}$. If $n = 0$, then $\hat{\varphi}$ is already quantifier-free. So there is nothing to do. Consider now the case $n > 0$. We then either have $Q_nx_n = \forall x_n$ or $Q_nx_n = \exists x_n$. The former case can be reduced to the latter one by means of the equivalence $\forall x_n\varphi \iff \neg\exists x_n\neg\varphi$. In the latter case we equivalently transform φ into elimination normal form $\bar{\varphi}$ with respect to x_n . By Lemma 12 there exists an elimination set E for $\exists x_n\bar{\varphi}$. That is

$$\text{PAI} \models \exists x_n\varphi \iff \exists x_n\bar{\varphi} \iff \bigvee_{(\gamma,t) \in E} \gamma \wedge \bar{\varphi}[t//x_n] \iff \bigvee_{(\gamma,t) \in E} \gamma \wedge \varphi[t//x_n].$$

We obtain $\hat{\varphi}^*$ from $\hat{\varphi}$ by equivalently replacing $\exists x_n\varphi$ with the last disjunction above, and we can eliminate the remaining quantifiers from $\hat{\varphi}^*$ by our induction hypothesis. □

Corollary 14. *PAI is substructure-complete, complete, and decidable.*

Proof. Admitting quantifier elimination is known to be equivalent to substructure completeness. For deciding an L_∞ -sentence in PAI we eliminate all quantifiers according to Theorem 13, then decide all atomic formulas in $\mathbb{Z} \cup \{\infty\}$, and finally obtain an equivalent truth value via propositional calculus. This exhibits also the completeness. □

5 Complexity

All complexity bounds discussed in the sequel are based on the assumption that the integers in the input formulas are binary coded. Notice that our formal language L_∞ would actually require to code them unary as sums of L_∞ -constants 1 possibly preceded by the unary function symbol “−.” Taking this into account, however, might improve the bounds by one exponential step but would not appropriately describe practical computations.

Notice that from a complexity point of view it is absolutely essential to have ternary predicate symbols for the congruences. When considering, in contrast, countably infinitely many binary ones—one for each modulus—our procedure is not elementary recursive as by increasing moduli the number of test points can be arbitrarily increased without increasing the input length. Again, our choice of ternary predicate symbols, where the (logarithm of the) size of the modulus contributes to the input length, establishes an appropriate model for practical computations.

Our write-up of our elimination procedure in the previous two sections is driven by the idea to use a small and natural language L_∞ and to separate mathematical foundations from implementation issues. Turning to complexity we have to discuss and revise one detail of this procedure: The application of Lemma 3(i) for making positive logically negated congruences introduces a number of atomic formulas, which is linear in the modulus. Since that modulus is represented in binary, this causes an exponential blow-up, which can however be easily avoided: one simply leaves negated congruences, which are directly preceded by logical negation, unchanged. One could call this a *weakly positive* formula. This is correct due to the following observation: The role of congruences in the elimination set computation in Lemma 9 is only that their modulus contributes to some least common multiple computation, and applying Lemma 3(i) would introduce several new congruences but no new moduli.

Since the size of our elimination set in Lemma 12 is essentially that of the part obtained via Lemma 9 for the regular Presburger case, we obtain the asymptotic upper bounds given by Weispfenning for a similar procedure 8. That is, the procedure is triply exponential in the input word length. More precisely, it is triply exponential in the number of quantifiers. This bound is known to be tight for regular Presburger arithmetic 4.

Weispfenning observed, however, that this result still can be improved in several ways 23,8. First, consider the elimination of several consecutive existential quantifiers in the proof of Theorem 13: When proceeding like

$$\exists x_{n-1} \exists x_n \varphi \longleftrightarrow \exists x_{n-1} \bigvee_{(\gamma,t) \in E} \gamma \wedge \varphi[t//x_n] \longleftrightarrow \bigvee_{(\gamma,t) \in E} \exists x_{n-1} (\gamma \wedge \varphi[t//x_n])$$

and independently eliminating $\exists x_{n-1}$ from several smaller subproblems, one achieves that test points originating from a certain subproblem are not substituted within other subproblems. An analogue observation holds with consecutive universal quantifiers. This way, the quantifier elimination procedure is triply

exponential in the number of quantifier alternations but only doubly exponential in the number of quantifiers for a bounded number of alternations.

Second, one can gain one exponential step by introducing big disjunction symbols “ \bigvee ” as an abbreviated notation for regular disjunctions following common mathematical practice. These big disjunctions can be used for substituting the big unions of elimination terms ranging over k in Lemma 9. The crucial observation is that this is compatible with the elimination procedure in the sense that these big disjunctions need not be expanded for the elimination of subsequent quantifiers. Our original main technical Lemma for *uniform* Presburger arithmetic [9, Lemma 8], is in fact a formalization of this approach, since for regular Presburger arithmetic the bounded quantifiers considered there represent such big disjunctions, and this approach is compatible with our extension considered here.

6 Elimination Examples

We start with two examples for quantifier elimination in PAI in order to illustrate the procedure and to give an intuition of the semantics. Then we turn to a more complex example pointing at one possible application of our work.

Example 15 (There is a maximum). Consider the sentence $\exists x \forall y (y \leq x)$, which in regular Presburger arithmetic is equivalent to “false.” We start with the elimination of the inner quantifier $\forall y$ using the equivalence $\forall y (y \leq x) \iff \neg \exists y (\neg y \leq x)$. We bring $\neg y \leq x$ into positive elimination normal form with respect to y by applying Lemma 3(ii), Lemma 7(ii), and Lemma 8:

$$\begin{aligned} \neg y \leq x &\iff x + 1 \leq y \wedge x - x = 0 \\ &\iff (x - y + 1 \leq 0 \vee y = \infty) \wedge x - x = 0 \\ &\iff ((-y \leq -x - 1 \wedge y - y = 0 \wedge x - x = 0) \vee y = \infty) \wedge x - x = 0. \end{aligned}$$

Applying Lemma 12 we compute $A = \{-y \leq -x - 1\}$ and obtain from Lemma 9 the following elimination set with respect to y :

$$E = \{(\text{true}, x), (\text{true}, x + 1), (\text{true}, x + 2)\}.$$

To simplify our discussion here, we observe that all guarded points except for $(\text{true}, x + 1)$ can be dropped from E without violating Remark 10. This yields

$$E' = \{(\text{true}, x + 1), (\text{true}, 0), (\text{true}, \infty)\}.$$

The application of this E' yields

$$\begin{aligned} \forall y (y \leq x) &\iff \neg \exists y (\neg y \leq x) \\ &\iff \neg \bigvee_{(\gamma, t) \in E'} \gamma \wedge ((x - y + 1 \leq 0 \vee y = \infty) \wedge x - x = 0)[t//y] \\ &\iff \neg (x - x = 0 \vee x + 1 \leq 0 \vee x - x = 0) \\ &\iff \neg x - x = 0. \end{aligned}$$

Now $\neg x - x = 0$ is equivalent to $x = \infty$ so that for the elimination of the outer quantifier $\exists x$ we obtain “true” via (true, ∞) , which is generally contained in our elimination sets. \square

Example 16 (There is a minimum). Consider the sentence $\exists x \forall y (x \leq y)$, which is dual to our previous example. Again we construct a positive elimination normal form with respect to y of $\neg x \leq y$ to eliminate the inner quantifier:

$$\neg x \leq y \iff y \leq x - 1 \wedge y - y = 0.$$

For $A = \{y \leq x - 1\}$ Lemma 9 essentially yields $\{(\text{true}, x - 1)\}$, and according to Lemma 12 we obtain the elimination set

$$E' = \{(\text{true}, x - 1), (\text{true}, 0), (\text{true}, \infty)\}.$$

The application of E' yields

$$\forall y (y \leq x) \iff \neg(x - x = 0 \vee 0 \leq x - 1 \vee \text{false}) \iff \text{false},$$

and it follows that $\exists x \forall y (y \leq x) \iff \text{false}$. \square

Recently, computer science research has focussed on decidable fragments of Boolean algebras of power sets with cardinality constraints where the interpretation of the 1 is a finite or countably infinite set [14][15]. This resulted in a theory called BAPA combining Boolean algebra and Presburger arithmetic in a two-sorted approach. The decision procedure for BAPA is based on quantifier elimination. It reduces an input BAPA formula to an L -formula in our sense. This method still yields a decision procedure when combined with any decidable extension of Presburger arithmetic. In particular the results in [14][15] are compatible with our extension PAI. We are now going to demonstrate by means of an example how PAI can be combined with Boolean algebras.

Example 17 (Boolean algebra with cardinality). Consider the following problem: We are looking for a non-empty set which is a subset of every infinite element of the powerset of some countable ground set. In the language of Boolean algebras with cardinality constraints a first-order formulation is given by

$$|A| > 0 \wedge \forall X (|X| = \infty \implies A \cap X = A).$$

Notice that we have $|X| = \infty$ as a constraint, which in BAPA is only first-order definable, e.g. $\exists k (|\overline{A}| = k)$. Notice furthermore that we do not have a decision problem but a quantifier elimination problem here.

We are going to transform the quantified part of the given formula into an L_∞ -formula. To start with,

$$\forall X (|X| = \infty \implies A \cap X = A) \iff \neg \exists X (|X| = \infty \wedge \overline{A} \cup X \neq 1).$$

We can equivalently replace $\overline{A} \cup X \neq 1$ by $A \cap \overline{X} \neq \emptyset$. The latter can be transformed into a cardinality constraint $|A \cap \overline{X}| > 0$. Analogously $|X| = \infty$ can be

expressed by $|X \cap A| + |X \cap \bar{A}| = \infty$. We additionally introduce two tautological constraints

$$|A| - |X \cap A| - |\bar{X} \cap A| = 0 \quad \text{and} \quad |A| - |X \cap \bar{A}| - |\bar{X} \cap \bar{A}| = 0.$$

We replace each cardinality by a variable ranging over $\mathbb{Z} \cup \{\infty\}$ as follows:

$$|A| = x, \quad |X \cap A| = a, \quad |\bar{X} \cap A| = b, \quad |X \cap \bar{A}| = c, \quad |\bar{X} \cap \bar{A}| = d.$$

This finally yields

$$\neg \exists a \exists b \exists c \exists d (x \geq 0 \wedge a \geq 0 \wedge b \geq 0 \wedge c \geq 0 \wedge d \geq 0 \wedge \\ x - a - b = 0 \wedge x - c - d = 0 \wedge a + c = \infty \wedge c > 0).$$

Eliminating the quantified variables a, b, c, d we would obtain the formula $x = 0$. The solution set with respect to x describes the cardinality of all sets A which satisfy our condition. In our case the result $x = 0$ contradicts the condition $|A| > 0$ outside the scope of the quantifier in the original problem. Consequently there is no such set A . \square

7 Conclusions and Further Work

We have given a quantifier elimination procedure and a corresponding decision procedure for Presburger arithmetic with infinity. The asymptotic worst-case complexity is not worse than that of corresponding procedures for regular Presburger arithmetic, which are widely accepted as an important and useful tool. The next reasonable step is to implement our procedure in the computer logic system REDLOG [29], which already features an implementation of quantifier elimination for regular Presburger Arithmetic [9,10], the essential part of which can be reused as a subroutine. One can then address quantifier elimination for Boolean algebras with cardinality constraints and examine the practical applicability of our approach to problems from deductive databases or software verification [13,14,15]. For Boolean algebras with cardinality constraints it is a promising idea to adopt for PAI the concept of *positive quantifier elimination* restricting to variables that describe positive or non-negative numbers including infinity. For applications of real quantifier elimination to problems in algebraic biology this brought a considerable progress [30]. Finally note that our approach can be expected to admit also *extended quantifier elimination*, where one obtains satisfying sample values for existentially quantified variables if such values exist [31].

Acknowledgment

We thank Peter Revesz, who pointed us at his work on deductive databases and quantifier elimination for certain Boolean algebras via reduction to Presburger formulas. Peter also pointed us at the option to extend this to considering infinite sets on the basis of a potential Presburger implementation with infinity.

References

1. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: *Comptes Rendus du premier congrès de Mathématiciens des Pays Slaves*, Warsaw, Poland, pp. 92–101 (1929)
2. Cooper, D.C.: Theorem proving in arithmetic without multiplication. *Machine Intelligence* 7, 91–99 (1972)
3. Ferrante, J., Rackoff, C.W.: *The Computational Complexity of Logical Theories*. Lecture Notes in Mathematics, vol. 718. Springer, Berlin (1979)
4. Fischer, M.J., Rabin, M.: Super-exponential complexity of Presburger arithmetic. *SIAM-AMS Proceedings* 7, 27–41 (1974)
5. Reddy, C.R., Loveland, D.W.: Presburger arithmetic with bounded quantifier alternation. In: *STOC 1978: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pp. 320–325. ACM, New York (1978)
6. Pugh, W.: The omega test: a fast and practical integer programming algorithm for dependence analysis. In: *Supercomputing 1991: Proceedings of the 1991 ACM/IEEE Conference on Supercomputing*, pp. 4–13. ACM, New York (1991)
7. Oppen, D.C.: A $2^{2^{2^n}}$ upper bound on the complexity of Presburger arithmetic. *J. Comput. Syst. Sci.* 16(3), 323–332 (1978)
8. Weispfenning, V.: The complexity of almost linear Diophantine problems. *Journal of Symbolic Computation* 10(5), 395–403 (1990)
9. Lasaruk, A., Sturm, T.: Weak quantifier elimination for the full linear theory of the integers. A uniform generalization of Presburger arithmetic. *Applicable Algebra in Engineering, Communication and Computing* 18(6), 545–574 (2007)
10. Lasaruk, A., Sturm, T.F.: Weak integer quantifier elimination beyond the linear case. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) *CASC 2007*. LNCS, vol. 4770, pp. 275–294. Springer, Heidelberg (2007)
11. Davis, M.: Final report on mathematical procedures for decision problems. Technical report, Institute for Advanced Study, Princeton, NJ (October 1954), Under Technical Supervision of Commanding General, Aberdeen Proving Ground. Work Performed During Period 1, to 31, Under Contract No. DA-36-034-ORD-1645. Department of Army Project No. 599-01-004 (1954)
12. Luckham, D.C., German, S.M., von Henke, F.W., Karp, R.A., Milne, P.W., Oppen, D.C., Polak, W., Scherlis, W.L.: *Stanford Pascal verifier user manual*. Technical report, Stanford University, Stanford, CA, USA (1979)
13. Revesz, P.Z.: Quantifier-elimination for the first-order theory of boolean algebras with linear cardinality constraints. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) *ADBIS 2004*. LNCS, vol. 3255, pp. 1–21. Springer, Heidelberg (2004)
14. Kuncak, V., Nguyen, H.H., Rinard, M.: An algorithm for deciding BAPA: Boolean algebra with presburger arithmetic. In: Nieuwenhuis, R. (ed.) *CADE 2005*. LNCS (LNAI), vol. 3632, pp. 260–277. Springer, Heidelberg (2005)
15. Kuncak, V., Nguyen, H.H., Rinard, M.: Deciding Boolean algebra with Presburger arithmetic. *Journal of Automated Reasoning* 36(3), 213–239 (2006)
16. Kuncak, V.: Quantifier-free Boolean algebra with Presburger arithmetic is NP-complete. Technical Report TR-2007-001, MIT Computer Science and AI Lab, Cambridge, MA (January 2007)
17. Weispfenning, V.: Quantifier elimination and decision procedures for valued fields. In: Mueller, G.H., Richter, M.M. (eds.) *Models and Sets*. Proceedings of the Logic Colloquium held in Aachen, July 18–23, 1983 Part I. Lecture Notes in Mathematics (LNM), vol. 1103, pp. 419–472. Springer, Heidelberg (1984)

18. Monk, J.D.: *Mathematical Logic*. Graduate Texts in Mathematics, vol. 37. Springer, Heidelberg (1976)
19. Löwenheim, L.: Über Möglichkeiten im Relativkalkül. *Mathematische Annalen* 76(4), 447–470 (1915)
20. Skolem, T.: Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theoreme über dichte Mengen. *Videnskapsselskapet Skrifter, I. Matematisk-naturvidenskabelig Klasse* 6, 1–36 (1920)
21. Malcev, A.: Untersuchungen aus dem Gebiete der mathematischen Logik. *Rec. Math. [Mat. Sbornik] N.S.* 1(43)(3), 323–336 (1936)
22. Dolzmann, A., Sturm, T.: Simplification of quantifier-free formulae over ordered fields. *Journal of Symbolic Computation* 24(2), 209–231 (1997)
23. Weispfenning, V.: The complexity of linear problems in fields. *Journal of Symbolic Computation* 5(1&2), 3–27 (1988)
24. Loos, R., Weispfenning, V.: Applying linear quantifier elimination. *The Computer Journal* 36(5), 450–462 (1993); Special issue on computational quantifier elimination
25. Weispfenning, V.: Quantifier elimination for real algebra—the quadratic case and beyond. *Applicable Algebra in Engineering Communication and Computing* 8(2), 85–101 (1997)
26. Sturm, T.: Linear problems in valued fields. *Journal of Symbolic Computation* 30(2), 207–219 (2000)
27. 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*, Institut für Informatik, Technische Universität München, Garching, Germany, pp. 285–300 (2002)
28. Dolzmann, A.: *Algorithmic Strategies for Applicable Real Quantifier Elimination*. Doctoral dissertation, Universität Passau, 94030 Passau, Germany (July 2000)
29. Dolzmann, A., Sturm, T.: Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin* 31(2), 2–9 (1997)
30. Sturm, T., Weber, A., Abdel-Rahman, E.O., El Kahoui, M.: Investigating algebraic and logical algorithms to solve Hopf bifurcation problems in algebraic biology. *Mathematics in Computer Science* 2(3), 493–515 (2009)
31. Weispfenning, V.: Simulation and optimization by quantifier elimination. *Journal of Symbolic Computation* 24(2), 189–208 (1997)

An Algorithm for Symbolic Solving of Differential Equations and Estimation of Accuracy

Natasha Malaschonok

Tambov State University,
Internatsionalnaya 33, 392622 Tambov, Russia

Abstract. An algorithm for solving systems of differential equations based on Laplace transform method is presented. There are considered ordinary linear differential equations with constant coefficients, nonzero initial conditions and right-hand sides as composite functions reducible to sums of exponents with polynomial coefficients.

An algorithm to compute an error of calculations sufficient to obtain a preassigned accuracy of solution of linear differential equations system is included.

Present-day computer systems provide an ample equipment for solving differential equations. For numerical methods, there are many algorithms to estimate an error of obtained approximate solutions. If it concerns symbolic algorithms one can hardly find such estimations as it is customary to presume an exact character of analytic solving. But nearly each symbolic algorithm of solving contains numerical components or is based on approximation of involved functions or other mathematical structures by series, products, sequences, etc. It is necessary to guarantee an adequate accuracy in this case as well.

An algorithm, which is presented in this article, is based on the application of the Laplace transform method for solving the systems of differential equations. This method provides a symbolic character of computations. However, there exists a fragment of numerical calculations. It concerns the computing of polynomial roots. This is the operation that requires an estimation of accuracy for calculations. An algorithm for such estimation is presented in this paper.

The Laplace transform has been very useful in various problems of differential equations theory (see, for example, [BurgHall05](#), [DahSN99](#), [MY97](#), [Pod97](#)).

In this article, we consider systems of ordinary linear differential equations with constant coefficients, nonzero initial conditions, and right-hand sides as composite functions reducible to the sums of exponents with polynomial coefficients. A case of continuous right-hand sides was discussed in [NMa105a](#), [NMa105b](#). Here we consider a general case of composite functions and obtain estimates in a general case and of higher accuracy. Such systems are very important because of their application for many problems in electronics, electrical or radio engineering, economics, etc. Moreover such systems provide a transparent application of Laplace method.

In the first section of the paper, an algorithm for solving systems of differential equations is presented. At first a preparation of data functions for the formal Laplace transform is performed. It is achieved with application of Heaviside function and moving the obtained functions into the bounds of smoothness intervals. The next step is solving the algebraic system with polynomial coefficients and the right-hand side obtained after the Laplace transform of the data system. There are algorithms that are efficient for solving this type of equations, and are different for various types of such systems. Then the obtained solution of algebraic system is prepared to the inverse Laplace transform. It is reduced to the sum of partial fractions with exponential coefficients. Just at this stage it is necessary to calculate an error of the denominator roots sufficient for the required accuracy of differential equations solutions.

The second section is devoted to an estimation of an error of calculations sufficient to obtain a preassigned accuracy of solution of a system of differential equations. We must underscore that it is not necessary to obtain the exact solution of the system of differential equations in order to obtain the value of an error sufficient for the required accuracy of an approximate solution.

The complexity of computations is discussed in the third section.

In the last section, an example is considered.

1 An Algorithm for Laplace Transform Method of Solving Systems of Differential Equations

Consider [\[NMa106\]](#) the system

$$\sum_{j=1}^n \sum_{k=0}^N a_{kj}^l x_j^{(k)} = f_l, \quad l = 1, \dots, n, \quad a_{kj}^l \in \mathbf{R}, \tag{1}$$

of n differential equations of order N with initial conditions $x_j^{(k)}(0) = x_{0j}^k$, $k = 0, \dots, N - 1$ and right-hand functions f_l reduced to the form

$$f_l(t) = f_l^i(t), \quad t_l^i < t < t_l^{i+1}, \quad i = 1, \dots, I_l, \quad t_l^1 = 0, \quad t_l^{I_l+1} = \infty, \tag{2}$$

where

$$f_l^i(t) = \sum_{s=1}^{S_l^i} P_{ls}^i(t) e^{b_{ls}^i t}, \quad i = 1, \dots, I_l, \quad l = 1, \dots, n,$$

and $P_{ls}^i(t) = \sum_{m=0}^{M_{ls}^i} c_{sm}^{li} t^m$.

We denote by $x_j, j = 1, \dots, n$ the unknown functions of argument $t, t \geq 0$, and by $x_j^{(k)}$ the order k derivative of the function $x_j, k = 0, \dots, N$.

An algorithm suggested here will be disposed in three stages.

Stage 1. The Laplace transform of the input system

Denote the Laplace images of the functions $x_j(t)$ and $f_l(t)$ by $X_j(p)$ and $F_l(p)$, respectively.

Step 1.1. The Laplace transform of the left-hand side of system (1) with respect to the initial conditions is performed by formal writing the expression

$$\sum_{j=1}^n \sum_{k=0}^N a_{kj}^l p^k X_j(p) - \sum_{j=1}^n \sum_{k=0}^{N-1} d_{jk}^l(p) x_{0j}^k,$$

where

$$d_{jk}^l(p) = \sum_{i=k}^{N-1} a_{i+1,j}^l p^{i-k}. \tag{3}$$

Step 1.2. The preparation of right-hand functions $f_l(t)$ to the Laplace transform is reduced to the application of Heaviside function $\eta(t)$. Represent $f_l(t)$ as a sum

$$f_l^m(t) = \phi_l^m(t - t_l^k) = \sum_{s=1}^{S_l^m} \psi_{ls}^m(t - t_l^k) e^{b_{ls}^m t_l^k} e^{b_{ls}^m(t - t_l^k)}.$$

Here $\psi_{ls}^m(t - t_l^k) = P_{ls}^m(t)$ and $\psi_{ls}^m(t - t_l^k) = \sum_{j=0}^{J_{ls}^m} \gamma_{lsj}^{mk} (t - t_l^k)^j$. Coefficients γ_{lsj}^{mk} are calculated by the formula

$$\gamma_{lsj}^{mk} = \sum_{r=0}^{M_{ls}^k - j} c_{s,m+r}^{lk} \binom{m+r}{r} (t_l^k)^r.$$

Finally the function $f_l(t)$ is reduced to the form

$$f_l(t) = \sum_{i=1}^{I_l-1} [\phi_l^i(t - t_l^i) \eta(t - t_l^i) - \phi_l^i(t - t_l^{i+1}) \eta(t - t_l^{i+1})] + \phi_l^{I_l}(t - t_{I_l}) \eta(t - t_{I_l}).$$

Step 1.3. Since the Laplace image of $(t - t^*)^n e^{\alpha(t - t^*)} \eta(t - t^*)$ is $\frac{n!}{(p - \alpha)^{n+1}} e^{-t^* p}$ the Laplace transform of $f_l(t)$ is the following:

$$F_l(p) = \sum_{i=1}^{I_l-1} [\Phi_l^{i,i}(p) - \Phi_l^{i,i+1}(p)] + \Phi_l^{I_l, I_l}(p). \tag{4}$$

For each $l = 1, \dots, n$ we reduce (4) to the common denominator. The common denominator is left factorized. At that the numerator is the sum of exponents with polynomial coefficients.

Stage 2. The solution of the linear algebraic system with polynomial coefficients

Consider the linear algebraic system of the order n with polynomial coefficients relative to $X_j, j = 1, \dots, n$:

$$\sum_{j=1}^n \sum_{k=0}^N a_{kj}^l p^k X_j(p) = \sum_{j=1}^n \sum_{k=0}^{N-1} d_{jk}^l(p) x_{0j}^k + F_l(p), \quad l = 1, \dots, n. \tag{5}$$

Step 2.1. With notation

$$\sum_{j=1}^n \sum_{k=0}^{N-1} d_{jk}^l(p) x_{0j}^k = S_l(p).$$

system (5) is reduced to the form

$$\sum_{j=1}^n \sum_{k=0}^N a_{kj}^l p^k X_j(p) = S_l(p) + F_l(p), \quad l = 1, \dots, n. \tag{6}$$

For each $l = 1, \dots, n$ the expressions on the right-hand side of (5) are reduced to the common denominator.

Step 2.2. System (6) may be solved by any possible classical method, for example, Cramer’s method. But now there are developed new effective procedures, for example, p-adic method [Mal03], modula methods, and methods based on determinant identities [Ma97], [Ma00].

Stage 3. The inverse Laplace transform

Step 3.1. The solution of (6), i.e., each desired function $X_j(p)$, $j = 1, \dots, n$, is represented as a fraction with polynomial denominator. This denominator is partially factored – it contains the multipliers of $F_l(p)$ denominators and the determinant $D(p)$ of system (6). The numerator is the sum of exponents with polynomial coefficients. We reduce the function $X_j(p)$, $j = 1, \dots, n$, to the sum of exponents with fractional coefficients. Numerators and denominators of these coefficients are polynomials.

The main step is the decomposition of each fraction in the $X_j(p)$ expansion into the sum of partial fractions $A/(p - p^*)^v$, $p^* \in \mathbf{C}$. The roots of $D(p)$ must be determined.

Step 3.2. The determination of the $D(p)$ roots may be performed in various ways, for example by reducing to the system of two equations on \mathbf{R} with respect to $p_1 = \text{Re}p$, $p_2 = \text{Im}p$.

This step includes an algorithm of calculation of accuracy. This algorithm will be described further in **Section 2**.

Step 3.3. Preparation for inverse Laplace transform is the decomposition of rational fractions or fractional coefficients of exponents into the sums of partial fractions $A/(p - p^*)^v$, $p^* \in \mathbf{C}$. A standard method, for example, is the method of indefinite coefficients. It is based on solving the system of algebraic equations over \mathbf{C} . It may be solved as it was mentioned already by modula or p-adic method.

Each $X_j(p)$ is finally represented as a sum

$$X_j(p) = \sum_m \sum_k \frac{A_{mk}}{(p - p_{jk})^{\beta_{mk}}} e^{-\alpha_m p}. \tag{7}$$

Step 3.4. The Laplace originals of functions $X_j(p)$ are obtained formally – by writing the expressions

$$x_j(t) = \sum_m \sum_k \frac{A_{mk}}{(\beta_{mk} - 1)!} (t - \alpha_m)^{\beta_{mk} - 1} e^{p_{ik}(t - \alpha_m)} \eta(t - \alpha_m), j = 1, \dots, n.$$

Remark. In general, the functions $\tilde{x}_l(t)$ are complex valued. We take the real part of $\tilde{x}_l(t)$ for each $l = 1, \dots, n$. The functions $\text{Re} \tilde{x}_l(t)$ may be taken as the solution of system (1), i.e., the required functions $x_l(t)$. It is easy to show that the error would not exceed the established precision assured by the calculated accuracy of roots of $D(p)$.

2 Estimation of Accuracy

At **Step 3.2** of Laplace method, it is necessary to calculate the roots of $D(p)$. The problem is to compute an error of roots sufficient to the desired accuracy differential equations solutions. In this section, the algorithm for this computation is constructed.

We shall consider all functions and make calculations on the segment $[0, T]$, where $T > t_l^l$ for all $l = 1, \dots, n$, and is sufficiently high for the input problem. Denote by $\tilde{x}_l(t)$ an approximate solution of (1) constructed by means of approximate roots of $D(p)$. We require the following accuracy for solutions on the segment $[0, T]$:

$$\max_{t \in [0, T]} |x_l(t) - \tilde{x}_l(t)| < \varepsilon, \quad l = 1, \dots, n. \tag{8}$$

We must determine an error Δ of the $D(p)$ roots sufficient for the required accuracy ε for $x_l(t)$. For the case when the right-hand sides of (1) are continuous see ([\[NMa105a\]](#) – [\[NMa105b\]](#)).

Here we consider a general case of composite right-hand sides.

Denote by $T_i^l(p)$ the i, l minor of the matrix of system (6). The solution $X_l(p)$ of system (6) may be expressed in the following way:

$$X_l(p) = \frac{D_l(p)}{D(P)},$$

where

$$D_l(p) = \sum_{i=1}^m [F_l(p) + S_l(p)] T_i^l(p).$$

Denote by p_r roots, and by $p_r^*, r = 1, \dots, n$, approximate roots of the polynomial $D(p)$. If the error of a root is less than Δ , then $|p_r - p_r^*| < \Delta$.

Theorem. *For every ε there exists such Δ that if $|p_r - p_r^*| < \Delta$, then (8) is true.*

Proof. Let us consider the polynomial $D(p + \Delta e^{i\alpha})$, $\alpha \in [0, 2\pi]$. Denote

$$\tilde{X}_l(p) = \frac{D_l(p)}{D(p + \Delta e^{i\alpha})}. \tag{9}$$

We must find Δ which produces (8).

For (8) and (9) we must estimate the original of

$$\frac{D_l(p)}{D(p)} - \frac{D_l(p)}{D(p + \Delta e^{i\alpha})}.$$

According to the linearity of Laplace and inverse Laplace transforms we estimate separately the Laplace originals of

$$\sum_{i=1}^n F_l(p) \frac{T_i^l(p)}{D(p)} - \sum_{i=1}^n F_l(p) \frac{T_i^l(p)}{D(p + \Delta e^{i\alpha})} \tag{10}$$

and

$$\sum_{i=1}^n S_l(p) \frac{T_i^l(p)}{D(p)} - \sum_{i=1}^n S_l(p) \frac{T_i^l(p)}{D(p + \Delta e^{i\alpha})}. \tag{11}$$

Write expression (10) in the form:

$$\sum_{i=1}^n F_l(p) \left(\frac{T_i^l(p)}{D(p)} - \frac{T_i^l(p)}{D(p + \Delta e^{i\alpha})} \right). \tag{12}$$

Both functions $F_l(p)$ and the difference in the brackets have the Laplace originals. The original $\Psi_l(t)$ of (10) is calculated as a sum of convolutions of $f_l(t)$ and the original $\Omega_l^i(t)$ of

$$\frac{T_i^l(p)}{D(p)} - \frac{T_i^l(p)}{D(p + \Delta e^{i\alpha})}. \tag{13}$$

Let us present (13) as follows:

$$\frac{T_i^l(p)}{D(p)} - \frac{T_i^l(p + \Delta e^{i\alpha})}{D(p + \Delta e^{i\alpha})} + \frac{T_i^l(p + \Delta e^{i\alpha}) - T_i^l(p)}{D(p + \Delta e^{i\alpha})}. \tag{14}$$

Denote by $Q_i^i(t)$ and $\widetilde{Q}_i^i(t)$ the originals of $\frac{T_i^l(p)}{D(p)}$ and $\frac{T_i^l(p)}{D(p)} - \frac{T_i^l(p + \Delta e^{i\alpha})}{D(p + \Delta e^{i\alpha})}$, correspondingly. Then according to the properties of Laplace transform we have:

$$\widetilde{Q}_i^i(t) = (1 - e^{\Delta T}) Q_i^i(t). \tag{15}$$

Now let us estimate $Q_i^i(t)$. The function $Q_i^i(t)$ may be written in the form

$$Q_i^i(t) = \sum_{r=1}^R \left(\sum_{\mu=1}^{\mu_r} \mathcal{B}_{r\mu}^i \frac{t^{\mu_r - \mu}}{(\mu_r - \mu)!} \right) e^{p_r t}, \tag{16}$$

where p_r are the roots of $D(p)$ of order μ_r , R is the amount of various roots.

Coefficients $\mathcal{B}_{r\mu}^i$ may be calculated as the Taylor coefficients of the functions $(p - p_r)^{\mu_r} \frac{T_i^l(p)}{D(p)}$ at the point p_r . They may be estimated using the Cauchy inequality for the Taylor coefficients in the δ -neighborhood of the point p_r . Let be $\delta < 1/2 \min |p_r - p_s|, r, s = 1, \dots, R, \delta < 1$. We wish to obtain $\Delta \leq \delta$. If it would be not so, we shall take $\Delta = \delta$.

According to Cauchy evaluations we have:

$$|\mathcal{B}_{r\mu}^i| \leq \left(\max_{|p-p_r|=\delta} (p - p_r)^{\mu_r} \frac{T_i^l(p)}{D(p)} \right) / \delta^\mu.$$

Consider the polynomial $D(p) = \sum_{\nu=0}^{mn} c_\nu p^\nu$. As

$$\left| (p - p_r)^{\mu_r} \frac{T_i^l(p)}{D(p)} \right| = \frac{|T_i^l(p)|}{|c_{mn}| \prod_{s=1, s \neq r}^R |p - p_r|^{\mu_s}}$$

we need to estimate $|T_i^l(p)|$ in the δ -neighborhood of p_r . Denote by ρ the radius of the circle containing all the roots of $D(p)$. For such ρ we may take the number

$$\rho = \max \left\{ 1; \frac{\sum_{\nu=0}^{mn} |c_\nu|}{|c_{mn}|} \right\},$$

it may be chosen such that

$$\max_{|p-p_r|=\delta} |T_i^l(p)| \leq \max_{|p|=\rho} |T_i^l(p)|.$$

Each element of the matrix, whose determinant is denoted by $T_i^l(p)$, is a polynomial $\gamma_{\eta,\lambda}(p)$, where λ is a number of its row, $\lambda \neq l, \eta$ is a number of its column, $\eta \neq i$. Denote $m_{\eta,\lambda} = \max_{|p|=\rho} |\gamma_{\eta,\lambda}(p)|$. According to Hadamard inequality

$$\max_{|p|=\rho} |T_i^l(p)| \leq \sqrt{\prod_{\eta=1, \eta \neq i}^n \sum_{\lambda=1, \lambda \neq l}^n m_{\eta,\lambda}^2}.$$

Denote

$$M_{il} = \sqrt{\prod_{\eta=1, \eta \neq i}^n \sum_{\lambda=1, \lambda \neq l}^n m_{\eta,\lambda}^2}. \tag{17}$$

As $|p - p_s| \geq \delta$, taking into account that $\mu \leq |mu_r$, we get:

$$|\mathcal{B}_{r\mu}^i| \leq \frac{M_{il}}{|c_{mn}| \delta^{mn}}.$$

As a result, we obtain the estimate for $Q_i^i(t)$ on the segment $[0, T]$:

$$|Q_i^i(t)| \leq \frac{mn}{|c_{mn}| \delta^{mn}} M_{il} e^{\rho T}. \tag{18}$$

Consider the last item of (14). Due to the mean value theorem

$$T_i^l(p) - T_i^l(p + \Delta e^{i\alpha}) = \Delta e^{i\alpha} \frac{1}{2\pi i} \int_{|\xi-p|=\delta} \frac{T_i^l(\xi)}{(\xi - p - \Delta e^{i\alpha})(\xi - p)} d\xi. \tag{19}$$

From (19) we obtain the inequality:

$$|T_i^l(p) - T_i^l(p + \Delta e^{i\alpha})| \leq \frac{\Delta}{\delta - \Delta} M_{il}. \tag{20}$$

As previously for the original $\widetilde{\widetilde{Q}}_i^l(t)$ of $\frac{T_i^l(p+\Delta e^{i\alpha})-T_i^l(p)}{D(p+\Delta e^{i\alpha})}$ we obtain the following estimate:

$$\left| \widetilde{\widetilde{Q}}_i^l(t) \right| \leq \frac{\Delta}{\delta - \Delta} |Q_i^l(t)|. \tag{21}$$

From (15), (18), (20), and (21) we estimate $\Omega_i^l(t)$:

$$|\Omega_i^l(t)| \leq \left(e^{\Delta T} - 1 + \frac{\Delta}{\delta - \Delta} \right) \frac{mn}{|c_{mn}| \delta^{mn}} M_{il} e^{\rho T}. \tag{22}$$

The original $f_i(t)$ of $F_i(p)$ is composite. Each component of $f_i(t)$ is a sum of exponents with polynomial coefficients. It is estimated with respect to maximums of coefficients at the corresponding segments and values of exponents at the right-hand ends of the segments. Denote

$$\max_{t \in [0, T]} |f_i(t)| = M f_i. \tag{23}$$

Evaluating the convolution of originals for (12) we obtain the estimate for $\Psi_l(t)$:

$$\max_{t \in [0, T]} |\Psi_l(t)| \leq (e^{\Delta T} - 2) \frac{mnT}{|c_{mn}| \delta^{mn}} \sum_{i=1}^n M f_i M_{il} e^{\rho T}.$$

Denote

$$\Lambda_l = \frac{mnT}{|c_{mn}| \delta^{mn}} \sum_{i=1}^n M f_i M_{il} e^{\rho T}. \tag{24}$$

Consider (11). The polynomial $S_i(p)$ has no original. Denote

$\sum_{i=1}^n S_i(p) T_i^l(p) = K_l(p)$, $\frac{K_l(p)}{D(p)} = N_l(p)$. Then (11) may be written as

$$\begin{aligned} \frac{K_l(p)}{D(p)} - \frac{K_l(p)}{D(p + \Delta e^{i\alpha})} &= \frac{K_l(p)}{D(p)} - \frac{K_l(p + \Delta e^{i\alpha})}{D(p + \Delta e^{i\alpha})} + \\ \Delta e^{i\alpha} \frac{1}{2\pi i} \int_{|\xi-p|=\delta} &\frac{N_l(p)}{(\xi - p - \Delta e^{i\alpha})(\xi - p)} d\xi. \end{aligned} \tag{25}$$

For the original $R_l(t)$ of $N_l(p)$ the representation like (16) is possible. Evaluations analogues to previous reduce to the estimations:

$$\max_{t \in [0, T]} |R_l(t)| \leq \frac{mnT}{|c_{mn}| \delta^{mn}} \sum_{i=1}^n \max_{|p| \leq \rho} |S_i(p)| M_{il} e^{\rho T}. \tag{26}$$

From (5) it is evident that

$$\max_{|p| \leq \rho} |S_i(p)| \leq \sum_{\nu=k}^{m-1} |a_{\nu+1,j}^i| \rho^{\nu-k} |x_{oj}^k|. \tag{27}$$

Denote

$$L_l = \frac{mnT}{|c_{mn}| \delta^{mn}} \sum_{i=1}^n \max_{|p| \leq \rho} |S_i(p)| M_{il} e^{\rho T}. \tag{28}$$

Similarly to previous evaluations we obtain for the original $\Theta_l(t)$ of (11):

$$|\Theta_l(t)| \leq (e^{\Delta T} - 1 + \frac{\Delta}{\delta - \Delta}) L_l. \tag{29}$$

Finally, from (22) and (29) we obtain the following estimation for $x_l(t)$:

$$\max_{t \in [0, T]} |x_l(t) - \tilde{x}_l(t)| \leq \left(e^{\Delta T} - 1 + \frac{\Delta}{\delta - \Delta} \right) (A_l + L_l). \tag{30}$$

We shall suppose that $\frac{k}{k+1} \delta < \Delta < \delta, k \geq 0$. Then inequality (30) looks like

$$\max_{t \in [0, T]} |x_l(t) - \tilde{x}_l(t)| \leq (e^{\Delta T} - 1 + k) (A_l + L_l). \tag{31}$$

Starting from (9) and (31) we require

$$(e^{\Delta T} - 1 + k) (A_l + L_l) < \varepsilon. \tag{32}$$

From (32) we find the estimation for Δ for each l , denote it by Δ_l :

$$\Delta_l < \frac{1}{T} \ln \left(\varepsilon (A_l + L_l)^{-1} + 1 - k \right). \tag{33}$$

We shall take

$$k < \min_l (A_l + L_l)^{-1}. \tag{34}$$

Note that calculations in [\[NMa05a\]](#), [\[NMa05b\]](#) implied $k = 0$.

Finally, we choose $\Delta = \min_l \Delta_l$.

The theorem is proved.

According to the theorem construct an algorithm for estimation.

Stage 4. Algorithm for estimation of accuracy

Step 4.1. Consider the matrix of system (6), calculate its submatrices and their elements the polynomials $\gamma_{\eta,\lambda}(p)$. Calculate $m_{\eta,\lambda} = \max_{|p|=\rho} |\gamma_{\eta,\lambda}(p)|$ and M_{il} (see (17)).

Step 4.2. Calculate Mf_i (see (23)).

Step 4.3. Calculate Δ_l (see (24)) according to values of δ and ρ as it was considered.

Step 4.4. Calculate L_l (see (28)), evaluating the polynomials $S_l(p)$ as in (27).

Step 4.5. The final result Δ_l and $\Delta = \min_l \Delta_l$ according to (33) and (34).

3 On the Complexity of the Algorithm

The complexity of the algorithm depends upon the complexity of three main operations: solving the system with polynomial coefficients at step 2.2, solving the system with constant coefficients for a representation in partial fractions at step 3.3, determination of the polynomial roots at step 3.2.

The right-hand side of system (6) at step 2.2 is the linear combination of exponents, the coefficients of which are the rational fractions. So step 2.2 is reduced mainly to the following operations: calculation of inverse polynomial matrix, multiplying polynomial matrices by polynomial vectors. The complexity for various algorithms of these calculations so as for solving systems at step 3.3 is estimated in [Mal91]. The fastest method for solving such systems is p-adic method ([Ma03]). The best for parallel machine is the modula method based on Chinese Remainder Theorem.

The complexity for step 3.2 may be found, for example, in [Akr89].

4 Example

Data system

$$\begin{cases} x_1''' - x_1' - 2x_1 - x_2''' + x_2 = f_1 \\ 3x_1''' + x_1'' - 2x_1' + x_2''' + x_2 = f_2. \end{cases}$$

Functions f_1 and f_2 and numbers t_i^j :

$$\begin{aligned} f_1^1 &= e^t, & f_1^2 &= t^2 e^{2t}, & t_1^1 &= 0, & t_1^2 &= 1; \\ f_2^1 &= te^t, & f_2^2 &= e^{2t}, & t_2^1 &= 0, & t_2^2 &= 1. \end{aligned}$$

Initial conditions:

$$x_{01}^0 = 5, \quad x_{01}^1 = 10, \quad x_{01}^2 = 30, \quad x_{02}^0 = 4, \quad x_{02}^1 = 14, \quad x_{02}^2 = 20.$$

Step 1.1.

$$\begin{aligned} &-2X_1 - pX_1 + p^3X_1 + X_2 - p^3X_2 - (10 - 4p - 4p^2 + 5(-1 + p^2)); \\ &-2pX_1 + p^2X_1 + 3p^3X_1 + X_2 + p^3X_2 - (110 + 14p + 4p^2 + 10(1 + 3p) + \\ &\qquad\qquad\qquad 5(-2 + p + 3p^2)). \end{aligned}$$

Step 1.2.

$$f_1 := (f_1^2 - f_1^1)\text{UnitStep}[(t - t_1^2)] + f_1^1\text{UnitStep}[t];$$

$$f_2 := (f_2^2 - f_2^1)\text{UnitStep}[(t - t_2^2)] + f_2^1\text{UnitStep}[t].$$

Step 1.3.

$$F_1 = \frac{1}{-1+p} - \frac{e^{1-p}}{-1+p} + \frac{e^{2-p}(p^2-2p+2)}{(-2+p)^3}; F_2 = \frac{e^{2-p}}{-2+p} + \frac{1}{(-1+p)^2} - \frac{e^{1-p}p}{(-1+p)^2}.$$

Step 2.1.

$$-2X_1 - pX_1 + p^3X_1 + X_2 - p^3X_2 =$$

$$10 - 4p - 4p^2 + 5(-1 + p^2) + \frac{1}{-1+p} - \frac{e^{1-p}}{-1+p} + \frac{e^{2-p}(p^2-2p+2)}{(-2+p)^3};$$

$$-2pX_1 + p^2X_1 + 3p^3X_1 + X_2 + p^3X_2 =$$

$$110 + 14p + 4p^2 + 10(1 + 3p) + 5(-2 + p + 3p^2) +$$

$$\frac{e^{2-p}}{-2+p} + \frac{1}{(-1+p)^2} - \frac{e^{1-p}p}{(-1+p)^2}.$$

Step 2.2.

$$X_1(p) =$$

$$e^{-p}(8e + 2e^2 - 4ep - 4e^2p + 2ep^2 + 2e^2p^2 + 9ep^3 - 6e^2p^3 - 19ep^4 +$$

$$12e^2p^4 + 11ep^5 - 8e^2p^5 - 2ep^6 + 2e^2p^6)/((-2 + p)^3(-1 + p)(-2 + p -$$

$$p^2 - 4p^3 - 3p^4 + p^5 + 4p^6)) +$$

$$(-856 + 1692p - 982p^2 + 1061p^3 - 1991p^4 + 1398p^5 - 412p^6 + 160p^7 -$$

$$95p^8 + 20p^9)/((-2 + p)^3(-1 + p)(-2 + p - p^2 - 4p^3 - 3p^4 + p^5 + 4p^6));$$

$$X_2(p) =$$

$$e^{-p}(-8e^2 - 32ep + 24e^2p + 64ep^2 - 28e^2p^2 - 32ep^3 + 17e^2p^3 - 24ep^4 - 5e^2p^4 +$$

$$34ep^5 - 3e^2p^5 - 14ep^6 + 5e^2p^6 + 2ep^7 - 2e^2p^7)/((-2 + p)^3(-1 + p)^2(-2 +$$

$$p - p^2 - 4p^3 - 3p^4 + p^5 + 4p^6)) +$$

$$(1776 - 4576p + 3568p^2 - 1404p^3 + 2465p^4 - 2751p^5 + 841p^6 + 133p^7 +$$

$$2p^8 - 68p^9 + 16p^{10})/((-2 + p)^3(-1 + p)^2(-2 + p - p^2 - 4p^3 - 3p^4 + p^5 + 4p^6)).$$

Step 3.1. $D(p) = -2 + p - p^2 - 4p^3 - 3p^4 + p^5 + 4p^6$.

To be short we produce the final results on the error of the roots. For arbitrary T :

Step 4.3. $\Lambda_1 = (6240T^3 + 20160T^2)e^{5T}$, $\Lambda_2 = (6240T^3 + 6720T^2)e^{5T}$.

Step 4.4.

$$L_1 = 9438240e^{4T}; L_2 = 3874080e^{4T}.$$

We shall take two variants of k .

Let $k = \frac{1}{2} \min_l (\Lambda_l + L_l)^{-1}$. The following table demonstrates the values of Δ according to ε and T :

$T \setminus \varepsilon$	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$
$T = 2$	$8.06 \cdot 10^{-13}$	$8.06 \cdot 10^{-14}$	$7.99 \cdot 10^{-15}$
$T = 3$	$6.22 \cdot 10^{-15}$	$6.66 \cdot 10^{-16}$	$7.14 \cdot 10^{-17}$

Let $k = \frac{1}{10} \min_l (A_l + L_l)^{-1}$. Then we obtain the corresponding table:

$T \setminus \varepsilon$	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$
$T = 2$	$1.45 \cdot 10^{-12}$	$1.45 \cdot 10^{-13}$	$1.44 \cdot 10^{-14}$
$T = 3$	$1.12 \cdot 10^{-14}$	$1.11 \cdot 10^{-15}$	$1.48 \cdot 10^{-16}$

We shall take $\Delta = 10^{-15}$ and calculate approximate roots of $D(p)$.

$$\begin{aligned}
 p_1^* &= -1.000000000000000, p_2^* = -0.594937842169665 - i0.830713582043548, \\
 p_3^* &= -0.594937842169665 + i0.830713582043548, p_4^* = 0.355937297682321 - \\
 & i0.513128324882554, p_5^* = 0.355937297682321 + i0.513128324882554, p_6^* = \\
 & 1.22800108897469.
 \end{aligned}$$

The decomposition into the sum of partial fractions is rather bulky, so we omit some steps and produce the final result – the solution of the system of differential equations .

$$\begin{aligned}
 x_1(t) &= tt10.031249e^{-t} - 1.25e^t + 5.538602e^{1.228001t} + \\
 & 2e^{0.355937t}(-3.735568\text{Cos}0.513128t + 15.529795\text{Sin}0.513128t) + \\
 & 2e^{-0.594937t}(-0.924357\text{Cos}0.830713t + 0.061193\text{Sin}0.830713t); \\
 x_2(t) & 10.03125e^{-t} + 0.5e^t - 8.948223e^{1.228001t} + 0.5e^t t + \\
 & 2e^{0.355937t}(-0.493116\text{Cos}0.513128t + 33.959275\text{Sin}0.513128t) + \\
 & 2e^{-0.594937t}(1.701602\text{Cos}0.830713t + 0.929609\text{Sin}0.830713t).
 \end{aligned}$$

5 Conclusion

Let us adduce advantages of the algorithm presented in the paper.

1. The Laplace transform is a way for symbolic solving of differential equations as it reduces the solution process to algebraic manipulations.
2. Representation of righthand side functions by sums of exponents with polynomial coefficients (in a case when it is possible) makes the Laplace transform completely symbolic.
3. The algebraic system obtained after the Laplace transform may be solved by methods most convenient and efficient for each specific case.
4. Decomposition of algebraic equations into a sum of partial fractions with exponential coefficients provides a symbolic character of the inverse Laplace transform.
5. An algorithm to compute an error of calculations sufficient to obtain a preassigned accuracy of solution of a system of linear differential equations is presented. This method is realized according to the input data. It does not demand the exact solution of the system.

References

[Akr89] Akritas, A.G.: Elements of Computer Algebra with Applications. J. Wiley Interscience, New York (1989)

[BurgHall05] Burghlea, D., Haller, S.: Laplace transform, dynamics and spectral geometry (January 17, 2005) arXiv:math.DG/0405037v2

- [DahSN99] Dahiya, R.S., Saberi-Nadjafi, J.: Theorems on n-dimensional Laplace transforms and their applications. In: 15th Annual Conf. of Applied Math., Univ. of Central Oklahoma, Electr. Journ. of Differential Equations, Conf. 02, pp. 61–74 (1999)
- [DST87] Davenport, J., Siret, Y., Tournier, E.: *Calcul Formel. Systemes et Algorithmes de Manipulations Algebriques*. MASSON, Paris (1987)
- [GG99] Von zur Gathen, J., Gerhard, J.: *Modern Computer Algebra*. Cambridge University Press, Cambridge (1999)
- [Mal91] Malaschonok, G.I.: Algorithm for the solution of systems of Linear Equations in commutative rings. In: Mora, T., Traverso, C. (eds.) *Effective Methods in Algebraic Geometry*. Progress in Mathematics, vol. 94, pp. 289–298. Birkhauser, Basel (1991)
- [Mal97] Malaschonok, G.I.: Recursive Method for the Solution of systems of Linear Equations. In: *Computational Mathematics, 15th IMACS World Congress, Berlin, August 1997, vol. I*, pp. 475–480. Wissenschaft und Technik Verlag, Berlin (1997)
- [Mal00] Malaschonok, G.I.: Effective Matrix Methods in Commutative Domains. In: *Formal Power Series and Algebraic Combinatorics*, pp. 506–517. Springer, Berlin (2000)
- [Mal03] Malaschonok, G.I.: Solution of systems of linear equations by the p-adic method. *Programming and Computer Software* 29(2), 59–71 (2003)
- [NMal05a] Malaschonok, N.: An algorithm to settle the necessary exactness in Laplace transform method. In: *Computer Science and Information Technologies, Berlin*, pp. 453–456 (2005)
- [NMal05b] Malaschonok, N.: Estimations in Laplace transform method for solutions of differential equations in symbolic computations. In: *Differential Equations and Computer Algebra Systems, Minsk*, pp. 195–199 (2005)
- [NMal06] Malaschonok, N.: Parallel laplace method with assured accuracy for solutions of differential equations by symbolic computations. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) *CASC 2006*. LNCS, vol. 4194, pp. 251–260. Springer, Heidelberg (2006)
- [MY97] Mizutani, N., Yamada, H.: Modified Laplace transformation method and its applications to anharmonic oscillator, February 12 (1997)
- [Pod97] Podlubny, I.: The Laplace transform method for linear differential equations of the fractional order, October 30 (1997) arXiv:funct-an/9710005v1

Lazy and Forgetful Polynomial Arithmetic and Applications

Michael Monagan¹ and Paul Vrbik²

¹ Simon Fraser University, Department of Mathematics, Burnaby, B.C. Canada

² The University of Western Ontario, Department of Computer Science, London, ON Canada

Abstract. We present lazy and forgetful algorithms for multiplying and dividing multivariate polynomials. The lazy property allows us to compute the i -th term of a polynomial without doing the work required to compute all the terms. The forgetful property allows us to forget earlier terms that have been computed to save space. For example, given polynomials A, B, C, D, E we can compute the exact quotient $Q = \frac{A \times B - C \times D}{E}$ without explicitly computing the numerator $A \times B - C \times D$ which can be much larger than any of A, B, C, D, E and Q . As applications we apply our lazy and forgetful algorithms to reduce the maximum space needed by the Bareiss fraction-free algorithm for computing the determinant of a matrix of polynomials and the extended Subresultant algorithm for computing the inverse of an element in a polynomial quotient ring.

1 Introduction

Lazy algorithms were first introduced into computer algebra systems by Burge and Watt [3] where they were used in Scratchpad II for power series arithmetic. But not all of the lazy power-series algorithms were efficient. For example, the most obvious algorithm for computing $\exp(f(x))$ to $O(x^n)$ requires $O(n^3)$ arithmetic operations whereas the lazy algorithm in [3] required $O(n^4)$. In [9] Watt showed how to reduce this to $O(n^2)$.

van der Hoeven considers lazy algorithms for multiplication of power series to $O(x^n)$ which are asymptotically fast [8]. A lazy analogue of Karatsuba's divide and conquer algorithm is given which does $O(n^{\log_2 3})$ arithmetic operations (the same as the as non-lazy algorithm) but uses $O(n \log n)$ space, an increase of a factor of $\log n$. van der Hoeven also gives a lazy multiplication based on the FFT which does $O(n \log^2 n)$ arithmetic operations, a factor of $\log n$ more than the non-lazy multiplication. However, all of these results assume *dense* power series and our interest is the sparse case.

Let D be an integral domain and $R = D[x_1, x_2, \dots, x_n]$ be a polynomial ring. Let $f = f_1 + f_2 + \dots + f_n$ be a polynomial in R where each term f_i of f is of the form $f_i = a_i X_i$ where $a_i \in D$ and X_i is a monomial in x_1, \dots, x_n . Two terms $a_i X_i, a_j X_j$ are *like terms* if $X_i = X_j$. We say f is in *standard form* if $a_i \neq 0$ and $X_1 \succ X_2 \succ \dots \succ X_n$ in a monomial ordering \succeq . This form is often called

the *sparse distributed form* for polynomials in R . In what follows we use $\#f$ to indicate the number of terms of f .

Let f, g be polynomials in the standard form. Johnson’s [5] multiplication algorithm is based on the observation that multiplying $f = f_1 + \dots + f_n$ by $g = g_1 + \dots + g_m$ can be done by executing a simultaneous m -ary merge on the set of *sorted* sequences

$$S = \{(f_1g_1, \dots, f_n g_1), \dots, (f_1g_m, \dots, f_n g_m)\}.$$

Johnson used a heap H , initialized to contain the terms $f_1g_1, f_1g_2, \dots, f_1g_m$, to merge the m sequences. The number of terms in this heap never exceeds $\#g$ and inserting into and extracting terms from H costs $O(\log \#g)$ monomial comparisons per insertion/extraction. Therefore, since all $\#f\#g$ terms are eventually inserted and extracted from the heap, the algorithm does a total of $O(\#f\#g \log \#g)$ monomial comparisons and requires auxiliary space for at most $\#g$ terms in the heap plus space for the output.

Monagan and Pearce [6] extended this heap algorithm to polynomial division. Recall that when we do $f \div g$ we are trying to construct the quotient q and remainder r such that $f - qg - r = 0$. One could use a heap to store the sum $f - qg$ by merging the set of $\#g + 1$ sorted sequences

$$\{(f_1, \dots, f_n), (-q_1g_1, \dots, -q_k g_1), \dots, (-q_1g_m, \dots, -q_k g_m)\}$$

where $m = \#g$ and $k = \#q$. Alternatively we may see the heap as storing the sum $f - \sum_{i=1}^m g_i \times (q_1 + q_2 + \dots + q_k)$.

These heap algorithms dealt only with the so-called *zealous* (non-lazy) polynomials. Our contributions are the variations of these algorithms that enable us to compute in a *lazy and forgetful* manner.

2 Lazy Arithmetic

The intended purpose of working in a lazy way is to improve performance by avoiding unnecessary calculations. To apply this to polynomial arithmetic we restrict access to a polynomial to that of a single term. Furthermore, we save intermediate results from this calculation so that the i -th term where $i \leq n$ will be ‘calculated’ instantaneously.

Definition 1. A lazy polynomial, F , is an approximation of the polynomial $f = f_1 + \dots + f_n$ (in standard form), given by $F^N = \sum_{i=1}^N f_i$ where $N \geq 0$. To ensure F^N is always defined we let $f_i = 0$ when $i > n$. This admits the useful notation $F^\infty = f$.

The terms F_1, \dots, F_N are called the *forced terms* of F and the nonzero terms of $f - F^N$ are called the *delayed terms* of F . We denote the number of forced terms of a lazy polynomial F by $|F|$ (and to be consistent let $\#F = |F^\infty| = \#f$).

A lazy polynomial must satisfy two conditions regarding computation: all the forced terms of F are cached for re-access and calculating a delayed term of F should force as few terms as possible.

Let us refine our focus and address the problem of determining the n -th term of a polynomial when it is the result of some operation. We will use the heap methods for division and multiplication and a simple merge for addition. Since these methods build the result in \succeq -order anyway, we simply halt and return once n non-zero terms are generated. But, in order to initially populate the heap one polynomial must be fully forced. We give an optimization that avoids this.

Claim. Let f, g be polynomials in the standard form and $S[j] = (f_1g_j, \dots, f_ng_j)$. If f_1g_j is in the heap H , then no term of the sequences $S[j + 1], \dots, S[m]$ can be the \succ -largest term of H .

Proof. By the definition of a monomial ordering we have: if $g_j \succ g_{j+1} \succ \dots \succ g_m$, then $f_1g_j \succ f_1g_{j+1} \succ \dots \succ f_1g_m$. As $f_1g_{j+1}, \dots, f_1g_m$ are (respectively) the \succeq -largest terms of $S[j + 1], \dots, S[m]$, it follows that f_1g_j is \succeq -larger than any term of $S[j + 1], \dots, S[m]$. The claim is an immediate consequence of this.

This claim gives a natural replacement scheme that ensures no term is prematurely calculated and put in the heap. For multiplication this is reflected in lines (13)-(15) of Algorithm 2. For division we replace a term coming out of the heap with the \succeq -next largest term in the sequence it was taken from. That is, we replace f_i with f_{i+1} and $-q_i g_j$ with $-q_{i+1} g_j$ (we also use the optimization that says only add $-q_{i+1} g_j$ after removing $-q_i g_j$). However, it is possible that we remove $-q_{i-1} g_j$ before q_i is known, in which case we would not be able to insert the term $-q_i g_j$. But, since $-q_i g_j$ can certainly not be required to calculate q_i , the terms needed to determine q_i must already be in the heap. Therefore, we can just remember the terms that should have been added to the heap, and eventually add them once q_i has been calculated. In the lazy division algorithm, this is referred to as ‘sleeping’.

We also require no work to be repeated to calculate X_{N-1}, \dots, X_1 after calculating X_N . To achieve this we pass our algorithms the approximation X^N , which must also record the state of the algorithm that generated it. Specifically, it must remember the heap the calculation was using and local variables that would otherwise get erased (we will assume that this information is associated with X^N in *some* way and can be retrieved and updated).

Lazy algorithms for doing multiplication and division are now presented. Note that the algorithm for division returns terms of the quotient (while updating the remainder), but could easily be modified to instead return terms of the remainder (while updating the quotient). Complexity results for multiplication and division follow their respective algorithms.

ALGORITHM 2 - LAZY MULTIPLICATION

Input: The lazy polynomials F and G so that $F^\infty = f$ and $G^\infty = g$, a positive integer N (the desired term), and the lazy polynomial X so that $X^\infty = f \times g$.

Output: The N -th term of the product $f \times g$.

- 1: **if** $N \leq |X|$ **then** $\{X_N$ has already been calculated. $\}$ **return** X_N ; **end if**
- 2: **if** $|X| = 0$ **then**
- 3: $\{X$ has no information. $\}$

```

4:   Initialize a heap  $H$  and insert  $(F_1G_1, 1, 1)$ ; {Order the heap by  $\succeq$  on the mono-
      mials in the first position.}
5:    $k \leftarrow 1$ ;
6:   else
7:     Let  $H$  be the heap associated with  $X$ ;
8:      $k \leftarrow$  number of elements in  $H$ ;
9:   end if
10:  while  $H$  is not empty do
11:     $t \leftarrow 0$ ;
12:    repeat
13:      Extract  $(s, i, j) \leftarrow H_{max}$  from the heap and assign  $t \leftarrow t + s$ ;
14:      if  $F_{i+1} \neq 0$  then Insert  $(F_{i+1}G_j, i + 1, j)$  into  $H$ ; end if
15:      if  $i = 1$  and  $G_{j+1} \neq 0$  then Insert  $(F_1G_{j+1}, 1, j + 1)$  into  $H$ ; end if
16:      until ( $H$  is empty) or ( $t$  and  $H_{max}$  are not like terms);
17:      if  $t \neq 0$  then  $(X_k, k) \leftarrow (t, k + 1)$ ; end if
18:      if  $k = N$  then Associate the heap  $H$  with  $X$ ; return  $X_k$ ; end if
19:    end while
20:  Associate the (empty) heap  $H$  with  $X$ ;
21:  return 0;

```

Theorem 1. *To force every term of X (that is to completely determine the standard form of $f \times g$) in Algorithm 2, requires $O(\#f\#g \log \#g)$ monomial comparisons, space for a heap with at most $\#g$ terms, and space for $O(\#f\#g)$ terms of the product.*

Proof. Proceeding as in [7], the size of the heap is not effected by line 14, as this merely replaces the term coming out of the heap in line 13. The only place the heap can grow is on line 15, which is bounded by the number of terms of g . Therefore $O(\#g)$ space is required for the heap. Since the product $f \times g$ has at most $\#f\#g$ many terms it will require $O(\#f\#g)$ space.

Extracting/inserting from/to a heap with $\#g$ elements does $O(\log \#g)$ many monomial comparisons. As every term of the product passes through the heap, we do $O(\#f\#g)$ extractions/insertions totaling $O(\#f\#g \log \#g)$ monomial comparisons.

Remark 1. It is possible to improve multiplication so that the heap requires space for only $\min(\#f, \#g)$ terms and the number of monomial comparisons done is $O(\#f\#g \log \min(\#f, \#g))$. If $\#f < \#g$ and we could switch the order of the input (i.e. calculate $g \times f$ instead of $f \times g$) then the heap would be of size $\#f$. But we we may not know $\#f$ and $\#g!$ So, we must quote the worst case scenario in our complexities (in fact we will emphasize this by using $\max(\#f, \#g)$).

ALGORITHM 3 - LAZY DIVISION

Input: The lazy polynomials F and G so that $F^\infty = f$ and $G^\infty = g$, a positive integer N (the desired term), and the lazy polynomials Q and R so that $f = g \times Q^\infty + R^\infty$.

Output: The N -th term of the quotient from $f \div g$.

```

1: if  $F_1 = 0$  then return 0; end if
2: if  $N \leq |Q|$  then { $Q_N$  has already been calculated.} return  $Q_N$ ;
3: if  $|Q| = 0$  then

```

```

4:   { $Q$  has no information.}
5:   Initialize a new heap  $H$  and insert  $F_1$  into  $H$ ;
6:    $s \leftarrow 2$ ;
7: else
8:   Let  $H$  be the heap associated with  $Q$ ;
9: end if
10: while  $H$  is not empty do
11:    $t \leftarrow 0$ ;
12:   repeat
13:     Extract  $x \leftarrow H_{max}$  from the heap and assign  $t \leftarrow t + x$ ;
14:     if  $x = F_i$  and  $F_{i+1} \neq 0$  then
15:       Insert  $F_{i+1}$  into  $H$ ;
16:     else if  $x = G_i Q_j$  and  $Q_{j+1}$  is forced then
17:       Insert  $-G_i Q_{j+1}$  into  $H$ ;
18:     else if  $x = G_i Q_j$  and  $Q_{j+1}$  is delayed then
19:        $s \leftarrow s + 1$ ; {Sleep  $-G_i Q_{j+1}$ }
20:     end if
21:     if  $x = G_i Q_1$  and  $G_{i+1} \neq 0$  then Insert  $-G_{i+1} Q_1$  into  $H$ ; end if
22:   until ( $H$  is empty) or ( $t$  and  $H_{max}$  are not like terms)
23:   if  $t \neq 0$  and  $g_1 | t$  then
24:      $Q_{|Q|+1} \leftarrow t/G_1$ ; {Now  $Q_{|Q|+1}$  is a forced term.}
25:     for  $k$  from 2 to  $s$  do
26:       Insert  $-G_k \cdot t/G_1$  into  $H$ ; {Insert all terms that are sleeping into  $H$ }
27:     end for
28:   else
29:      $R_{|R|+1} \leftarrow t$ ; {Now  $R_{|R|+1}$  is a forced term.}
30:   end if
31:   if  $|Q| = N$  then Associate the heap  $H$  with  $Q$ ; return  $Q_N$ ; end if
32: end while
33: Associate the (empty) heap  $H$  with  $Q$ ;
34: return 0;

```

Theorem 2. *To force every term of Q and R (that is to completely determine q and r such that $f = g \times q + r$) in Algorithm 3 requires $O((\#f + \#q\#g) \log \#g)$ many monomial comparisons, space for a heap with $O(\#g)$ terms, and space for $O(\#q + \#r)$ terms of the solution.*

Proof. Proceeding as in [6], the size of the heap H , denoted $|H|$ is unaffected by lines 15 and 17 since these lines only replace terms coming out of the heap. Line 19 merely increments s and does not increase $|H|$. The only place where H can grow is line 21 in which a new term of g is added to the heap, this is clearly bounded by $\#g$. It is clear that we require $O(\#q + \#r)$ space to store the quotient and remainder.

All terms of f and $q \times g$ are added to the heap, which is $\#f + \#q\#g$ terms. Passing this many terms through a heap of size $\#g$ requires $O((\#f + \#q\#g) \log \#g)$ monomial comparisons.

3 Forgetful Arithmetic

We propose a variant to lazy polynomial arithmetic that has useful properties. Consider that the operations from the previous section can be composed to form polynomial expressions. For example, we could use lazy arithmetic to calculate the n -th term of say, $A \times B - C \times D$. When we do this we store the intermediate terms. But, if re-access was not required we could ‘forget’ these terms. A ‘forgetful’ operation is like a lazy operation but intermediate terms won’t be stored. Forgetful operations are potentially useful when expanding compounded polynomial expressions with large intermediate subexpressions.

We can make some straightforward modifications to our lazy algorithms to accomplish this forgetful environment. Essentially all that is required is the removal of lines that save terms to the solution polynomial (i.e. lines that look like $X_i \leftarrow \square$) and eliminating any references to previous terms (or even multiple references to a current term). To emphasize this change we will limit our access to a polynomial by way of a `next` command.

Definition 2. *For some lazy polynomial F and monomial order \succeq , the `next` command returns the \succeq -next un-calculated term of a polynomial (eventually returning only zeros).*

Remark 2. The `next` command satisfies: $\text{next}(F) \succ \text{next}(F) \succ \dots \succ \text{next}(F) = 0 = \text{next}(F) = \dots$ and $\text{next}(F) + \text{next}(F) + \text{next}(F) + \dots = F^\infty$.

Definition 3. *A forgetful polynomial is a lazy polynomial that is accessed solely via the `next` command. That is, intermediate terms of F are not stored and can only be accessed once. If the functionality to re-access terms is restored in any way (i.e. by caching any term but the current term in memory), F is no longer considered to be a forgetful polynomial. Thus, for a forgetful polynomial F , calculating F_{n+1} forfeits access to the terms F_1 through F_n , even if these terms have never been accessed.*

Although it would be ideal to have all of our forgetful routines take forgetful polynomials as input and return forgetful polynomials as output, this is not possible without caching previous results. Consider multiplication for instance. Assuming that we must multiply each term of f by each term of g and we are limited to single time access to terms, this task is impossible. For if we calculate f_1g_2 we cannot then calculate f_2g_1 and vice versa.

For the same reason our division algorithm can not accept a forgetful divisor as it must be repeatedly multiplied by terms of the quotient (thus the quotient can not be forgetful either). However, the dividend *can* be forgetful which is a highly desirable feature (see Section 5). The only ‘fully’ forgetful (forgetful input and output) arithmetic operation we can have is addition (although polynomial differentiation and scalar multiplication are also fully forgetful).

The variant of multiplication that takes as input lazy polynomials, returning a forgetful polynomial, is a trivial change to Algorithm 2. In this case all that must be done is to remove the ‘if’ statement on line 18 so that the \succeq -next,

instead of the N -th, term is returned. As this is not a significant change, we will not present an algorithm for forgetful multiplication. Division will take as input a forgetful dividend and lazy divisor returning a *fully forced* quotient and remainder.

Theorem 3. *When multiplying f by g the worst case storage complexity for forgetful multiplication is $O(\max(\#f, \#g))$ (the storage required for the heap).*

Proof. A quick inspection of Algorithm 2 will show that the only time a previous term of the product is used is on line 2 and line 18. In both cases the term is merely being *re-accessed* and is not used to compute a new term of the product. Since we do not store nor re-access terms of a forgetful polynomial, we can eliminate the storage needed to do this requiring only space for a heap with $\max(\#f, \#g)$ terms.

ALGORITHM 5 - FORGETFUL DIVISION

Input: A forgetful polynomial F and lazy polynomial G so that $F^\infty = f$ and $G^\infty = g$.

Output: The lazy polynomials Q and R so that $f = g \times Q^\infty + R^\infty$.

```

1:  $t_F \leftarrow \text{next}(F)$ ;
2: if  $t_F = 0$  then Set  $Q$  and  $R$  to zero; return  $Q$  and  $R$ ; end if
3: Initialize a new heap  $H$  and insert  $t_F$  into  $H$ ;
4:  $s \leftarrow 2$ ;
5: while  $H$  is not empty do
6:    $t \leftarrow 0$ ;
7:   repeat
8:     Extract  $x \leftarrow H_{max}$  from the heap and assign  $t \leftarrow t + x$ ;
9:     if  $x = t_F$  then
10:        $t_F = \text{next}(F)$ 
11:       if  $t_F \neq 0$  then
12:         Insert  $t_F$  into  $H$ ;
13:       end if
14:       else if  $x = G_i Q_j$  and  $Q_{j+1}$  is forced then
15:         Insert  $-G_i Q_{j+1}$  into  $H$ ;
16:       else if  $x = G_i Q_j$  and  $Q_{j+1}$  is delayed then
17:          $s \leftarrow s + 1$ ; {Sleep  $-G_i Q_{j+1}$ }
18:       end if
19:       if  $x = G_i Q_1$  and  $G_{i+1} \neq 0$  then
20:         Insert  $-G_{i+1} Q_1$  into  $H$ ;
21:       end if
22:     until ( $H$  is empty) or ( $t$  and  $H_{max}$  are not like terms)
23:     if  $t \neq 0$  and  $g_1 | t$  then
24:        $Q_{|Q|+1} \leftarrow t/G_1$ ; {Now  $Q_{|Q|+1}$  is a forced term.}
25:       for  $k$  from 2 to  $s$  do
26:         Insert  $-G_k \cdot t/G_1$  into  $H$ ; {Insert all terms that are sleeping into  $H$ }
27:       end for
28:     else
29:        $R_{|R|+1} \leftarrow t$ ; {Now  $R_{|R|+1}$  is a forced term.}
30:     end if
31:   end while
32: return  $Q$  and  $R$ ;

```

In its current form Algorithm 5 returns a fully forced quotient Q and remainder R . It is straightforward to modify this algorithm to return a forgetful remainder instead. We simply have line 29 return t instead of saving a term to the remainder and change line 32 to return 0 (for when terms of R have been exhausted). In the interest of space we will assume this modification has been done as:

ALGORITHM 6 - FORGETFUL DIVISION (WITH FORGETFUL REMAINDER)

Input: The forgetful polynomial F and lazy polynomial G so that $F^\infty = f$ and $G^\infty = g$.

Output: The lazy polynomial Q and forgetful polynomial R so that $f = g \times Q^\infty + R^\infty$.

Theorem 4. *In algorithm 6, when calculating $f \div g$ the space required (including space for the input) to force every term of the forgetful remainder R is:*

1. Space for a heap with $\#g$ terms.
2. Space for $\#q$ terms of the quotient.
3. Space for $\#g$ terms of the divisor.
4. Space for one term of the dividend f .

Proof.

1. As there has been no change to the division algorithm, Theorem 2 implies the heap has $\#g$ many terms.
2. To fully force every term of a lazy polynomial Q requires storage for $\#q$ many terms.
3. As G is a lazy polynomial that will be fully forced during the execution we require space to store $\#g$ many terms for the divisor.
4. As F is a forgetful polynomial we are restricted to only accessing one term from F at a time (where no previously calculated terms are cached). Therefore we only require space to store one term of f .

4 Implementation

We have implemented a C-library for doing lazy (and forgetful) arithmetic for polynomials with coefficients that are machine integers modulo p , for p some

Listing 1.1. The lazy polynomial structure

```

1  struct poly {
2      int N;
3      TermType *terms;
4      struct poly *F1;
5      struct poly *F2;
6      TermType (*Method)(int n, struct poly *F,
7                          struct poly *G, struct poly *H);
8      int state[6];
9      HeapType *Heap;
10 };
11 typedef struct poly PolyType;

```

machine prime. In our implementation we represent monomials as single machine integers (which allows us to compare and multiply monomials in one machine instruction). This representation, analyzed by Monagan and Pearce [6], is based on Bachmann and Schönemann’s scheme [1]. The C-structure we are using to represent a lazy polynomial is given below.

The variable N is the number of forced terms, and $F1$ and $F2$ are two other lazy polynomials which the procedure `Method` (among `ADD`, `MULT`, `DIVIDE`, and `DONE`) is applied to. As previously discussed `Method` requires three inputs, two lazy polynomials to operate on, and a third lazy polynomial where the solution is stored (and where the current heap can be found). The array `state[6]` is a place to put local variables that get erased but need to be maintained, and `Heap` is the heap which the procedure `Method` uses.

The procedure `Term` produces the n -th term of the lazy polynomial F , calculating it if necessary, enabling us to follow the pseudo-code given more directly as `Term(i, F) = Fi`.

Listing 1.2. Term

```

1 TermType Term (int n, PolyType *F) {
2     if (n>F->N) {
3         return F->Method(n, F->F1, F->F2, F);
4     }
5     return F->terms[n];
6 };

```

Table 1. Benchmarks for Maple’s SDMP package [7], Singular, and our lazy package on sparse examples

	$f \times g \pmod{503}$			$(fg) \div f \pmod{503}$		
	SDMP	Singular	Lazy	SDMP	Singular	Lazy
$f = (1 + x + y^2 + z^3)^{20}$ $g = (1 + z + y^2 + x^3)^{20}$	0.26	0.28	1.2	0.28	0.38	1.4
$f = (1 + x + y^3 + z^5)^{20}$ $g = (1 + z + y^3 + x^5)^{20}$	0.35	0.67	1.3	0.38	0.65	1.4
$f = (1 + x + y^3)^{100}$ $g = (1 + x^3 + y)^{100}$	2.2	1.1	10.8	2.5	2.04	11.1

Many details about the implementation have been omitted but we note that we have built a custom wrapper that interfaces the C-library with Maple (a non-trivial technical feat). This allows us to manipulate polynomials in a lazy way at the Maple level but do calculations at the C level.

Benchmarks are given in Table 1 where we see that calculating in a lazy/forgetful manner is 3 – 5 times slower than calculating directly with Monagan and Pearce’s SDMP package (see [7]) or Singular. Roman Pearce pointed out that this is because we are not using *chaining* in our heap implementations. Chaining is a technique where like terms are grouped together in a linked list to dramatically reduce the number of monomial comparisons in the heap operations. In [7], Monagan and Pearce show that chaining improves the performance of multiplication and division using heaps by a factor of 3 – 5.

5 Applications

We give two similar, but nonetheless independently important, applications of forgetful polynomial arithmetic: the Bareiss algorithm and the Subresultant algorithm. These algorithms both have a deficiency in that intermediate calculations can become quite large with respect to the algorithms output. By using forgetful operations we can bypass the need to explicitly store intermediate polynomials and thus reduce the operating space of the each algorithm significantly.

5.1 The Bareiss Algorithm

The Bareiss algorithm is ‘fraction free’ approach for calculating determinants due to Bareiss [2] who noted that the method was first known to Jordan. The algorithm does exact divisions over any integral domain to avoid fractions.

The Bareiss algorithm is given below. In the case where $M_{k,k} = 0$ (which prevents us from dividing by $M_{k,k}$ in the next step) it would be straightforward to add code (between lines 2 and 3) to find a non-zero pivot. For the purpose of this exposition we assume no pivoting is required.

ALGORITHM 6 - BAREISS ALGORITHM

Input: M an n -square matrix with entries over an integral domain \mathcal{D} .

Output: The determinant of M .

```

1:  $M_{0,0} \leftarrow 1$ ;
2: for  $k = 1$  to  $n - 1$  do
3:   for  $i = k + 1$  to  $n$  do
4:     for  $j = k + 1$  to  $n$  do
5:        $M_{i,j} \leftarrow \frac{M_{k,k}M_{i,j} - M_{i,k}M_{k,j}}{M_{k-1,k-1}}$ ; {Exact division.}
6:     end for
7:   end for
8: end for
9: return  $M_{n,n}$ 

```

The problem is the exact division in line 5. In the final division where the determinant $M_{n,n}$ is obtained by dividing by $M_{n-1,n-1}$ the dividend must be larger than the determinant. It is quite possible (in fact typical) that this

calculation (of the form $\frac{A \times B - C \times D}{E}$) produces a dividend that is *much* larger than the corresponding quotient and denominator. This final division can be the bottleneck of the entire algorithm.

Example 1. Consider the symmetric Toeplitz matrix with entries from the polynomial ring $\mathbb{Z}[x_1, x_2, \dots, x_9]$ generated by $[x_1, \dots, x_9]$,

$$\begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_9 \\ x_2 & x_1 & x_2 & \cdots & x_8 \\ x_3 & x_2 & x_1 & \cdots & x_7 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ x_9 & \cdots & x_3 & x_2 & x_1 \end{bmatrix}.$$

When calculating the determinant of this matrix using Bareiss’ algorithm the last division (in line 5 of Algorithm 6) will have a dividend of 128,530 terms, whereas the divisor and quotient will only have 427 and 6,090 terms respectively.

To overcome this problem we use forgetful arithmetic to construct the quotient of $\frac{A \times B - C \times D}{E}$ without explicitly storing $A \times B - C \times D$ (the forgetful algorithms were invented to do precisely this calculation).

Theorem 5. *Calculating $Q = \frac{A \times B - C \times D}{E}$ (an exact division) with forgetful operations requires space for at most $O(\max(\#A, \#B) + \max(\#C, \#D) + \#E + \#Q)$ terms at any one time.*

Proof. We have from Theorem 3 that the products $A \times B$ and $C \times D$ require at most $\max(\#A, \#B)$ and $\max(\#C, \#D)$ space, where the difference of these products requires $O(1)$ since it is merely a merge. As there is no remainder because the division is exact, the division algorithm will use $O(\#E + \#Q)$ storage by Theorem 2. Summing these complexities gives the desired result.

The implications of this theorem can be observed in Table 2 where we have measured the amount of memory used by our implementation of the Bareiss algorithm with forgetful polynomials. The table shows a linear relationship with the size of the input polynomials. For $n = 8$ the total space is reduced by a factor of $57184/832 = 68$ (compared to a Bareiss implementation that explicitly stores the quotient), which is significant.

5.2 The Extended Subresultant Algorithm

Given a UFD \mathcal{D} and non-constant polynomial $m \in \mathcal{D}[x]$, we can form the quotient ring $F[x]/\langle m \rangle$ where F is the fraction field of \mathcal{D} . When m is an irreducible element of $\mathcal{D}[x]$ (that is, there is no non-constant $t \in \mathcal{D}[x]$ such that $t \neq m$ and t divides m), this quotient ring will be a field. Of course, when working in fields it is natural to ask if there is a systematic way of finding inverses. The extended subresultant algorithm does this by finding $s, t \in \mathcal{D}[x]$ such that

Table 2. Let $Q = \frac{A \times B - C \times D}{E}$ be the division of line 5 of the Bareiss algorithm and $\alpha = \max(\#A, \#B) + \max(\#C, \#D)$. The following is a measurement of memory used by our implementation of the Bareiss algorithm using forgetful polynomials to calculate $M_{n,n}$ when given the Toeplitz matrix generated by $[x_1, \dots, x_7]$.

n	$\#A$	$\#B$	$\#C$	$\#D$	$\#E$	$\#A\#B + \#C\#D$	$\alpha + \#E + \#Q$	32-bit words	\succeq -comparisons
5	12	15	17	17	4	469	106	426	2817
6	35	51	55	55	12	4810	306	944	45632
7	35	62	70	70	12	7070	326	1462	70028
8	120	182	188	188	35	57184	832	3468	720696

$s \cdot u + t \cdot m = \text{Res}(u, m, x)$. In this case $\text{deg}_x(s) < \text{deg}_x(m)$ and the inverse of $u \in F[x]/\langle m \rangle$ is $s/\text{Res}(u, m, x)$.

Our interest is finding subresultants in $\mathcal{D}[x]$ and inverses in $F[x]$ when $\mathcal{D} = \mathbb{Z}$ or $\mathcal{D} = \mathbb{Z}[y, z, \dots]$. The Subresultant algorithm uses pseudo-division instead of ordinary division (which the regular Euclidean algorithm uses) to avoid computing with fractions in the fraction field F of \mathcal{D} . We recall the definition of pseudo-remainder and pseudo-quotient.

Definition 4. Let $f, g \in \mathcal{D}[x]$. The pseudo-quotient \tilde{q} and pseudo-remainder \tilde{r} are the ordinary quotient and remainder of $\alpha \times f$ divided by g where $\alpha = \text{lcoeff}_x(g)^{\delta+1}$ and $\delta = \text{deg}_x(f) - \text{deg}_x(g)$. Thus they satisfy $\alpha f = \tilde{q}g + \tilde{r}$.

One can show (e.g. see Ch. 2. of [4]) that \tilde{q} and \tilde{r} are elements of $\mathcal{D}[x]$. The extended Subresultant algorithm is given by Algorithm 7. The operations deg_x , prem , pquo , and lcoeff_x , stand for the degree in x , pseudo-remainder, pseudo-quotient and leading coefficient in x (respectively).

ALGORITHM 7 - EXTENDED SUBRESULTANT ALGORITHM

Input: The polynomials $u, v \in \mathcal{D}[x]$ where $\text{deg}_x(u) \geq \text{deg}_x(v)$ and $v \neq 0$.
Output: The resultant $r = \text{Res}(u, v, x) \in \mathcal{D}$ and $s, t \in \mathcal{D}[x]$ where $s \cdot u + t \cdot v = r$.

- 1: $(g, h) \leftarrow (1, -1)$;
- 2: $(s_0, s_1, t_0, t_1) \leftarrow (1, 0, 0, 1)$;
- 3: **while** $\text{deg}_x(v) \neq 0$ **do**
- 4: $d \leftarrow \text{deg}_x(u) - \text{deg}_x(v)$;
- 5: $(\tilde{r}, \tilde{q}) \leftarrow (\text{prem}(u, v, x), \text{pquo}(u, v, x))$; $\{\tilde{r}, \tilde{q}$ are computed simultaneously. $\}$
- 6: $u \leftarrow v$;
- 7: $\alpha \leftarrow \text{lcoeff}_x(v)^{d+1}$;
- 8: $(s, t) \leftarrow (\alpha \cdot s_0 - s_1 \cdot \tilde{q}, \alpha \cdot t_0 - t_1 \cdot \tilde{q})$;
- 9: $(s_0, t_0) \leftarrow (s_1, t_1)$;
- 10: $v \leftarrow \tilde{r} \div (-g \cdot h^d)$;
- 11: $(s_1, t_1) \leftarrow (s \div (-g \cdot h^d), t \div (-g \cdot h^d))$
- 12: $g \leftarrow \text{lcoeff}_x(u)$;
- 13: $h \leftarrow (-g)^d \div h^{d-1}$;
- 14: **end while**
- 15: $(r, s, t) \leftarrow (v, s_1, t_1)$;
- 16: **return** r, s, t ;

Table 3. Let \tilde{r}, \tilde{q} be from line 5 and $v, -g \cdot h^d$ be from line 10 of Algorithm 7. The following is a measurement of the memory used by our implementation of the extended subresultant algorithm using forgetful polynomials to calculate $\text{Res}(f, g, x_1)$ where $f = x_1^8 + \sum_{i=1}^5 (x_i + x_i^3), g = x_1^4 + \sum_{i=1}^5 x_i^2 \in \mathbb{Z}[x_1, \dots, x_5]$ at iteration n .

n	$\#\tilde{r}$	$\#\tilde{q}$	$\#v$	$\#(-g \cdot h^d)$	32-bit words	>-comparisons
1	29	7	29	1	236	137
2	108	6	108	1	154	953
3	634	57	634	1	2,672	75,453
4	14,692	2412	2,813	70	83,694	25,801,600

A bottleneck occurs when finding the pseudo-remainder on line 5. It can be easily demonstrated, especially when u and v are sparse polynomials in many variables, that \tilde{r} is very large relative to the dividend and quotient given by the division on line 10. In fact \tilde{r} can be much larger than the resultant $\text{Res}(u, v, x)$.

Example 2. Consider the two polynomials $f = x_1^6 + \sum_{i=1}^8 (x_i + x_i^3)$ and $g = x_1^4 + \sum_{i=1}^8 x_i^2$ in $\mathbb{Z}[x_1, \dots, x_9]$. When we apply the extended subresultant algorithm to these polynomials we find that in the last iteration, the pseudo-remainder \tilde{r} has 427,477 terms but the quotient v has only 15,071 (v is the resultant in this case).

To solve this problem we let the pseudo-remainder be a forgetful polynomial so that the numerator on line 10 does not have to be explicitly stored. This is accomplished by using Algorithm 5 since (when f and g regarded as univariate polynomials in x) calculating $\text{prem}(f, g, x)$ is equivalent to dividing $\alpha \times f$ by g using ordinary division with remainder. Table 3 shows the benefit of calculating in this manner. In the final iteration only a max $634+2412=3046$ terms will need to be explicitly stored to calculate a pseudo-remainder with 14,692 terms. Note, in order to implement this pseudo-division in the sparse distributed form, the monomial ordering used must satisfy $Yx^n \succ Zx^{n-1}$ for all monomials Y and Z that do not involve x .

Conclusion

We presented algorithms for lazy and forgetful polynomial arithmetic and two applications. These applications have demonstrated that the space complexity of the Bareiss algorithm and extended Subresultant algorithm can be significantly improved by using forgetful arithmetic, as proposed in this paper.

Acknowledgement. We wish to thank Dr. Jürgen Gerhard, Dr. Marc Moreno Maza, Dr. Eric Schost, and Roman Pearce for their contributions to this work.

References

1. Bachman, O., Schönemann, H.: Monomial representations for Gröbner bases computations. In: Proceedings of ISSAC, pp. 309–316. ACM Press, New York (1998)
2. Bareiss, E.F.: Sylvester's identity and multisstep integer-preserving Gaussian elimination. *J. Math. Comp.* 103, 565–578 (1968)
3. Burge, W.H., Watt, S.M.: Infinite structures in Scratchpad II. In: Davenport, J.H. (ed.) ISSAC 1987 and EUROCAL 1987. LNCS, vol. 378, pp. 138–148. Springer, Heidelberg (1989)
4. Geddes, K.O., Czapor, S.R., Labahn, G.: Algorithms for Computer Algebra. Kluwer Academic Publishers, Dordrecht (1992)
5. Johnson, S.C.: Sparse polynomial arithmetic. *ACM SIGSAM Bulletin* 8(3), 63–71 (1974)
6. Monagan, M.B., Pearce, R.: Polynomial Division using Dynamic Arrays, Heaps, and Packed Exponent Vectors. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2007. LNCS, vol. 4770, pp. 295–315. Springer, Heidelberg (2007)
7. Monagan, M., Pearce, R.: Sparse polynomial arithmetic using a heap. *Journal of Symbolic Computation - Special Issue on Milestones In Computer Algebra* (2008) (submitted)
8. van der Hoeven, J.: Relax, but don't be too lazy. *J. Symbolic Computation* 11(1-000) (2002)
9. Watt, S.M.: A fixed point method for power series computation. In: Gianni, P. (ed.) ISSAC 1988. LNCS, vol. 358. Springer, Heidelberg (1989)

On the Average Growth Rate of Random Compositions of Fibonacci and Padovan Recurrences

Nikita Gogin and Aleksandr Mylläri

University of Turku, Finland
alemio@utu.fi

Abstract. An integer sequence $\{t_n\}$ defined by the random recurrence $t_0 = 0$, $t_1 = 1$, $t_2 = 1$, $t_{n+1} = t_{n-\xi} + t_{n-1-\xi}$, $n \geq 2$, where the random variable ξ is equal to 0 or 1 with the probabilities p and q respectively, is called a random composition of Fibonacci and Padovan recurrences. We show that $\lim_{n \rightarrow \infty} \sqrt[n]{\mathbf{E}(t_n)}$ is equal to the greatest absolute value of the roots of the algebraic equation $\lambda^3 = p\lambda^2 + \lambda + q$.

1 Introduction

The Fibonacci numbers defined by

$$f_0 = 0, f_1 = 1, f_{n+1} = f_n + f_{n-1}, n \geq 1 \quad (1)$$

and the Padovan numbers defined by

$$p_0 = 0, p_1 = 1, p_2 = 1, p_{n+1} = p_{n-1} + p_{n-2}, n \geq 2 \quad (2)$$

both are widely known. It is also well-known that f_n and p_n both increase exponentially with $n \rightarrow \infty$ at the rates

$$\phi = \frac{1 + \sqrt{5}}{2} = \sqrt{1 + \sqrt{1 + \sqrt{1 + \dots}}} = 1.61803398 \dots \quad (\text{the golden ratio}) \quad (3)$$

and

$$\begin{aligned} \gamma &= \frac{(9 - \sqrt{69})^{\frac{1}{3}} + (9 + \sqrt{69})^{\frac{1}{3}}}{2^{\frac{1}{3}} 3^{\frac{2}{3}}} = \sqrt[3]{1 + \sqrt[3]{1 + \sqrt[3]{1 + \dots}}} \\ &= 1.32471795 \dots \quad (\text{the plastic number}) \end{aligned} \quad (4)$$

respectively, where

$$\phi^2 = \phi + 1, \gamma^3 = \gamma + 1. \quad (5)$$

Let now ζ be a random variable such that

$$\zeta = 1 \text{ or } -1 \text{ with equal probabilities } \frac{1}{2} \text{ (a case of the "balanced coin")}$$

and let us consider a random sequence defined by the random recurrence

$$t_0 = 0, t_1 = 1, t_{n+1} = \zeta \cdot t_n + t_{n-1}, n \geq 1. \tag{6}$$

In his pioneer work [1], Divakar Viswanath proved that almost surely

$$\sqrt[n]{|t_n|} \rightarrow \alpha = 1.13198824\dots, \quad n \rightarrow \infty, \tag{7}$$

where $\log \alpha$ is computed as the integral of the function $\frac{1}{4} \log \left(\frac{1+4m^4}{(1+m^2)^2} \right)$ with respect to some fractal measure on \mathbf{R}^+ .

In the article [2], Viswanath’s result is generalized for the case of an “unbalanced coin”, and in *Remark 1.2* the authors of the article mention their previous results on the so-called “average point of view” in this problem, i.e., the problem of calculating the limit $\lim_{n \rightarrow \infty} \sqrt[n]{\mathbf{E}(t_n)}$, (\mathbf{E} here stands for the mean value), which is called “the average growth rate” (a.g.r.) of the sequence $\{t_n\}$ [3].

In our article we are interested in the problem of calculating the a.g.r. of the random sequence defined as follows:

Let

$$\begin{aligned} \xi \text{ be a random variable such that } \xi = 0 \text{ or } 1 \text{ with the probabilities} & \tag{8} \\ p \text{ and } q = 1 - p \text{ respectively, where } 0 \leq p \leq 1, & \end{aligned}$$

and let us define a random sequence $\{t_n\}$ by

$$t_0 = 0, t_1 = 1, t_2 = 1, t_{n+1} = t_{n-\xi} + t_{n-1-\xi}, n \geq 2. \tag{9}$$

The problem in question is to find the average growth rate of such sequences.

Recurrence (9) shows that the random variable “switches” randomly between recurrences (1) and (2), so we shall refer to the resulting random sequence as a *random composition* of Fibonacci and Padovan recurrences.

In the beginning of Section 2, for simplicity and clarity we consider in detail the case of a “balanced coin”, and afterwards we give a brief overview of the question for an “unbalanced coin”. In both cases we use in fact the method of “random Fibonacci tree” suggested in [3] that allows us to find a linear recurrence equation for $\mathbf{E}(t_n)$ by means of quite elementary considerations.

2 The Fibonacci Tree and the a.g.r.

We begin with the case of a ”balanced coin”:

Let ξ and $\{t_n\}$ be the same as defined in (8) and (9) with $p = \frac{1}{2}$, and let G be a random matrix:

$$G = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \xi & 1 & 1 - \xi \end{pmatrix}. \tag{10}$$

Then if $t = (0 \ 1 \ 1)^T$, coordinates of the vector

$$\begin{pmatrix} t_n \\ t_{n+1} \\ t_{n+2} \end{pmatrix} = G_{n-1} \dots G_1 G_0 \cdot t \tag{11}$$

where matrices G_{n-1}, \dots, G_1, G_0 are selected randomly from the set $\{F, P\}$, where $F = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ and $P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, constitute three consequent terms of $\{t_n\}$.

Below are three possible examples of such random sequences:

```

0  1  1  2  2  3  5  5  8  13  13  26 ...
0  1  1  1  2  2  4  6  10 16  16  32 ...
0  1  1  1  2  2  4  4  6   8  14  14 ...
    
```

Now we are going to describe a "random Fibonacci tree" for $p = \frac{1}{2}$ which indicates all possible outcomes for $\{t_n\}$ (cf. [3]) Each node in this tree represents a

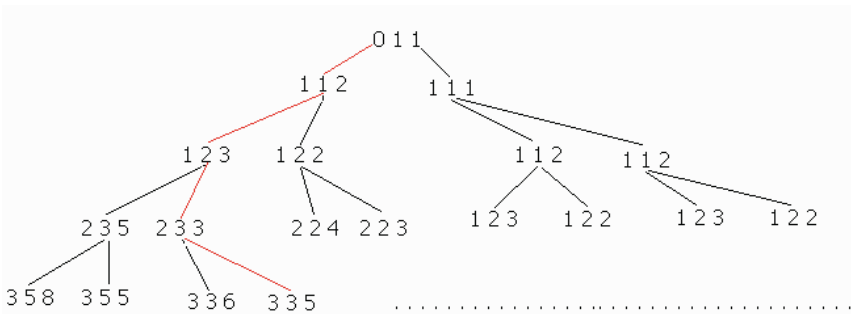


Fig. 1. The Fibonacci tree for the case of the balanced coin

parent-triple $(a = t_{n-2}, b = i_{n-1}, c = t_n)$ together with two children-triples: left $(b, c, b + c)$ and right $(b, c, a + b)$ in accordance with whether the parent-triple was multiplied by matrix F or P , respectively. So, any conceivable realization of $\{t_n\}$ corresponds to a (random) path in this tree. For example, the red path in Fig. 1 corresponds to a sequence $t_0 = 0, 1, 1, 2, 3, 3, 5, \dots$, where $t_n, n > 2$ is equal to the *third* element of a triple.

Let $(A_n, B_n, C_n), n \geq 0$ be a sum of all 2^n triples at the n th level of the tree. Then obviously $A_0 = 0, B_0 = 1, C_0 = 1$ and

$$\begin{aligned} A_{n+1} &= 2B_n, \quad B_{n+1} = 2C_n, \\ C_{n+1} &= (B_n + C_n) + (A_n + B_n) = A_n + 2B_n + C_n, \quad n \geq 0, \end{aligned} \tag{12}$$

hence $C_0 = 1, C_1 = 3, C_2 = 9$ and for $n \geq 2$

$$C_{n+1} = C_n + 2(2C_{n-1}) + 2(2C_{n-2}) = C_n + 4C_{n-1} + 4C_{n-2}. \tag{13}$$

From (13) we easily get a recurrence for the mean value $\mathbf{E}(C_n) = \mathbf{E}(t_n) = \frac{C_n}{2^n}$:

$$\begin{aligned} \mathbf{E}(t_0) = 1, \mathbf{E}(t_1) = \frac{3}{2}, \mathbf{E}(t_2) = \frac{9}{4}, \\ \mathbf{E}(t_{n+1}) = \frac{\mathbf{E}(t_n)}{2} + \mathbf{E}(t_{n-1}) + \frac{\mathbf{E}(t_{n-2})}{2}, n \geq 2. \end{aligned} \tag{14}$$

Since the characteristic equation for (14) is

$$2\lambda^3 = \lambda^2 + 2\lambda + 1, \tag{15}$$

the a.g.r. of the sequence $\{t_n\}$ is equal to the greatest absolute value of its roots:

$$\lambda_* = \left(1 + \sqrt[3]{73 + 6\sqrt{87}} + \sqrt[3]{73 - 6\sqrt{87}} \right) = 1.4375648970\dots \tag{16}$$

i.e. asymptotically

$$\mathbf{E}(t_n) \sim \lambda_*^n \text{ or } \log \mathbf{E}(t_n) \sim n \cdot \log \lambda_*. \tag{17}$$

Let now $p, 0 < p < 1$ be an arbitrary rational number, $p = \frac{m_1}{m_1+m_2}$, where m_1, m_2 are positive integers, and suppose one has a box containing m_1 and m_2 "black" = 0 and "white" = 1 enumerated balls respectively:

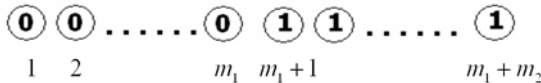


Fig. 2. A box containing m_1 "black" (0) and m_2 "white" (1) balls

So, the numbers $1, 2, \dots, m_1 + m_2$ being sampled equiprobably, the "black" and "white" balls will appear with probabilities p and $q = 1 - p = \frac{m_2}{m_1+m_2}$, respectively.

Let now $\{t_n\} = \{0 = t_0, 1, 1, t_3, t_4, t_5, \dots\}$, where

$$t_{n+1} = t_{n-\xi} + t_{n-1-\xi}, \quad n \geq 2 \tag{18}$$

and $\xi = 0$ or 1 for the "black" or "white" ball, respectively.

The construction of the Fibonacci tree for this case ("unbalanced coin") is quite similar to the previous one, but now every node of the tree is as shown in Figure 3.

For example, if $m_1 = 3, m_2 = 2$, the tree will look like the one in Figure 4.

Similarly to what was said above, let $(A_n, B_n, C_n), n \geq 0$ be the sum of all $(m_1 + m_2)^n$ triples at the n th level of this tree. Then $A_0 = 0, B_0 = 1, C_0 = 1$ and

$$\begin{aligned} A_{n+1} &= (m_1 + m_2)B_n, \quad B_{n+1} = (m_1 + m_2)C_n, \\ C_{n+1} &= m_2A_n + (m_1 + m_2)B_n + m_1C_n, \quad n \geq 0, \end{aligned} \tag{19}$$

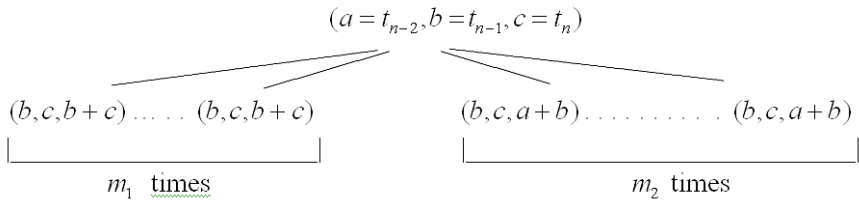


Fig. 3. A node of the Fibonacci tree for the case of unbalanced coin

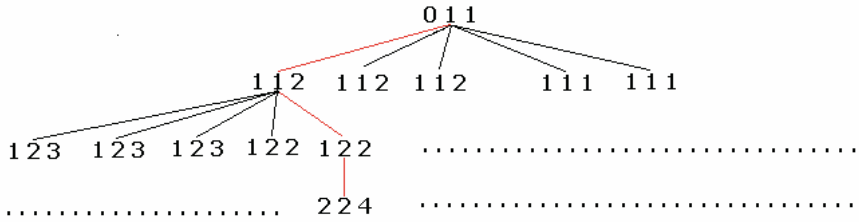


Fig. 4. An example of the Fibonacci tree for the case of unbalanced coin

from which it follows that

$$C_{n+1} = m_1 C_n + (m_1 + m_2)^2 C_{n-1} + m_2 (m_1 + m_2)^2 C_{n-2}, \quad n \geq 2 \quad (20)$$

$$C_0 = 1, C_1 = 2m_1^2 + m_2, C_2 = 3m_1 + 4m_1 m_2 + 2m_2^2.$$

Now for the mean value $\mathbf{E}(C_n) = \mathbf{E}(t_n) = \frac{C_n}{(m_1+m_2)^n}$ we get the recurrence relation:

$$\mathbf{E}(t_0) = 1, \mathbf{E}(t_1) = 1 + p, \mathbf{E}(t_2) = 2 + p^2, \quad (21)$$

$$\mathbf{E}(t_{n+1}) = p\mathbf{E}(t_n) + \mathbf{E}(t_{n-1}) + q\mathbf{E}(t_{n-2}), \quad n \geq 2$$

with characteristic equation

$$\lambda^3 = p\lambda^2 + \lambda + q. \quad (22)$$

By reason of continuity, equation (22) remains valid for any values of $p, 0 \leq p \leq 1$ rather than for only rational values.

So, for any $p, 0 \leq p \leq 1$ the a.g.r. of the sequence $\{t_n\}$ is equal to the greatest absolute value λ_* of the roots of equation (22) and again asymptotically

$$\log \mathbf{E}(t_n) \sim n \cdot \log \lambda_*. \quad (23)$$

Figure 5 shows a graphic of $\lambda_* = \lambda_*(p)$ for $p \in [0, 1]$, where $p = 0$ and $p = 1$ correspond to the Padovan and Fibonacci sequences, respectively:

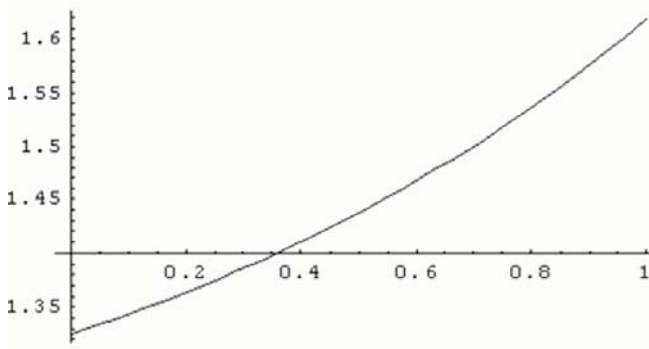


Fig. 5. A graphic of λ_* as a function of p

3 Computer Simulation Model

Figure 6 shows the result of the simulating by MATHEMATICA-program `FibPadCompose[N[Log[2]],7,50]`, where the random composition of the Fibonacci and Padovan recurrences with $p = \text{Log}[2] = 0.693147$ was generated for a sample of $M = 7$ sequences containing $N = 50$ terms in each sequence.

The seven ($= M$) *blue* curves represent the sequences $\{\log t_n\}_{n \leq N=50}$ whereas the *green* curve shows the logarithms of their "mean sequence"

$$\{\log \mathbf{E}(t_n)\}_{n \leq N=50},$$

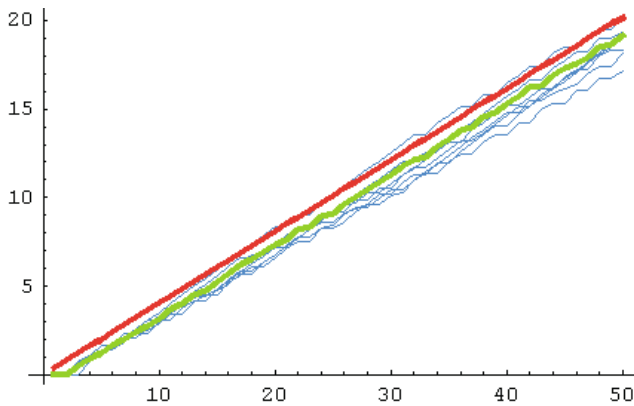


Fig. 6. An example of the simulating computer experiment for the random Fibonacci and Padovan composition

and the red straight line represents the asymptotic behavior of the $\{\log \mathbf{E}(t_n)\}_{n \leq N=50}$, i.e., according to (23) the sequence $\{n \cdot \log \lambda_*\}_{n \leq N=50}$. The slope of this line is equal to $\log \lambda_* = 0.403903\dots$ where $\lambda_* = 1.49766$.

It is evident that the green curve is very similar to a straight line parallel to the red one in full accordance with formula (23).

Acknowledgements

The authors are grateful to Dr. A. Farina (Italy, Rome) who attracted their attention to these problems [4]. A. Mylläri thanks also Vilho, Yrjö ja Kalle Väisälän Foundation (Vilho, Yrjö ja Kalle Väisälän rahasto) for the financial support.

References

1. Viswanath, D.: Random Fibonacci sequences and the number 1.13198824.. Math. Comp. 69(231), 1131–1155 (2000)
2. Janvresse, E., Rittaud, B., De La Rue, T.: How do random Fibonacci sequences grow? Probability Theory and Related Fields 142(3–4), 619–648 (2008)
3. Rittaud, B.: On the average growth of random Fibonacci sequences. J. Integer Sequences 10 (2007) Article 07.2.4
4. Benavoli, A., Chisci, L., Farina, A.: Fibonacci sequence, golden section, Kalman filter and optimal control. Signal Processing 89, 1483–1488 (2009)

A Study on Gröbner Basis with Inexact Input

Kosaku Nagasaka

Kobe University, Japan
nagasaka@main.h.kobe-u.ac.jp

Abstract. Gröbner basis is one of the most important tools in recent symbolic algebraic computations. However, computing a Gröbner basis for the given polynomial ideal is not easy and it is not numerically stable if polynomials have inexact coefficients. In this paper, we study what we should get for computing a Gröbner basis with inexact coefficients and introduce a naive method to compute a Gröbner basis by reduced row echelon form, for the ideal generated by the given polynomial set having a priori errors on their coefficients.

1 Introduction

Recently, computing a Gröbner basis for polynomials with inexact coefficients has been studied by several researchers ([1], [2], [3], [4], [5], [6], [7]). In Sasaki and Kako [1], this problem is classified into the first and the second kinds of problems. The first kind is computing a Gröbner basis for the ideal generated by the given polynomials with exact coefficients by numerical arithmetic (e.g. floating-point arithmetic). The second kind is for the given polynomials with inexact coefficients having a priori errors. In this case, we have to operate with a priori errors whether we compute a basis by exact arithmetic or not. For example, Shirayanagi's method ([3], [4]) by stabilization techniques requires to extend the input precision up to a point that the algorithm can work stably hence it is for the first kind since we cannot extend the input precision of inexact data in practice even if we can extend precisions during computations. For practical computations, coefficients may have a priori errors due to limited accuracy, representational error, measuring error and so on, hence the second kind is much more important than the first one. In this paper, we try to interpret the second kind of problem with the comprehensive Gröbner system and numerical linear algebra.

We assume that we compute a Gröbner basis or its variants for the ideal $I \subseteq \mathbf{C}[\mathbf{x}]$ generated by a polynomial set $F = \{f_1, \dots, f_k\} \subset \mathbf{C}[\mathbf{x}]$ where $\mathbf{C}[\mathbf{x}]$ is the polynomial ring in variables $\mathbf{x} = x_1, \dots, x_\ell$ over the complex number field \mathbf{C} . However, in our setting, coefficients may have a priori errors hence we have only a polynomial set $\tilde{F} = \{\tilde{f}_1, \dots, \tilde{f}_{\tilde{k}}\} \subset \mathbf{C}[\mathbf{x}]$ as the given inexact input, which may be different from F . We note that the number of polynomials may be also different (i.e. $k \neq \tilde{k}$). The most interesting part of this problem is what we should compute for the inexact input \tilde{F} when we are not able to discover the hidden and desirable polynomial set F . We review some known interpretation of this problem in Section 2 and 3 and give another resolution in the latter sections.

2 Comprehensive Gröbner System with Inexact Input

If we can bound the difference between F and \tilde{F} in some way, the most faithful solution for computing a Gröbner basis with inexact input is the comprehensive Gröbner basis (or comprehensive Gröbner system) introduced by Weispfenning ([8], [7], [9]). By representing error parts as unknown parameters, the problem becomes computing a parametric Gröbner basis. In this section, we briefly review this approach in our problem setting.

Let $\mathcal{A} = \mathbf{C}[\alpha_1, \dots, \alpha_\gamma]$ be the polynomial ring in parameters $\alpha_1, \dots, \alpha_\gamma$ over the complex number field and consider the polynomial ring $\mathcal{A}[\mathbf{x}]$ in variables x_1, \dots, x_ℓ . For a fixed term order \succ on $\mathbf{C}[\mathbf{x}]$, it is well-known that in general a Gröbner basis in $\mathcal{A}[\mathbf{x}]$ with respect to variables \mathbf{x} will no longer remain a Gröbner basis in $\mathbf{C}[\mathbf{x}]$ when the parameters $\alpha_1, \dots, \alpha_\gamma$ are specialized to some values in \mathbf{C} . The comprehensive Gröbner basis and system [8] are defined to overcome this situation.

Definition 1 (Comprehensive Gröbner Basis). *Let $F \subseteq \mathcal{A}[\mathbf{x}]$ be a finite parametric polynomial set and I be the ideal generated by F . We call a finite ideal basis G of I a comprehensive Gröbner basis of I if G is a Gröbner basis of the ideal generated by F in $\mathbf{C}[\mathbf{x}]$ for every specialization of parameters $\alpha_1, \dots, \alpha_\gamma$ in \mathbf{C} .* \triangleleft

Definition 2 (Comprehensive Gröbner System). *Let $F \subseteq \mathcal{A}[\mathbf{x}]$ be a finite parametric polynomial set, S be a subset of \mathbf{C}^γ , $\mathcal{A}_1, \dots, \mathcal{A}_r$ be algebraically constructible subsets of \mathbf{C}^γ such that $S \subseteq \mathcal{A}_1 \cup \dots \cup \mathcal{A}_r$ and G_1, \dots, G_r be subsets of $\mathcal{A}[\mathbf{x}]$. We call a finite set $G = \{(\mathcal{A}_1, G_1), \dots, (\mathcal{A}_r, G_r)\}$ of pairs a comprehensive Gröbner system for F on S if G_i is a Gröbner basis of the ideal generated by F in $\mathbf{C}[\mathbf{x}]$ for every specialization of parameters $(\alpha_1, \dots, \alpha_\gamma)$ in \mathcal{A}_i . Each (\mathcal{A}_i, G_i) is called a segment of G .* \triangleleft

Suppose that all the inexact parts on coefficients in \tilde{F} can be represented by parameters $\alpha_1, \dots, \alpha_\gamma$. Then, computing a Gröbner basis with inexact input can be done by computing a comprehensive Gröbner system for $F \in \mathcal{A}[\mathbf{x}]$ on S where S includes all the possible specialization of parameters $(\alpha_1, \dots, \alpha_\gamma)$ in \mathbf{C}^γ . However, in general, a comprehensive Gröbner system has a huge number of segments and its computation time is quite slow (see [10] for example). Though Weispfenning [7] tried to decrease the time-complexity by using only a single parameter to represent the inexact parts, whose bounding error mechanism is very similar to interval arithmetic and Traverso and Zanoni [6] pointed out that an interval easily becomes too large when we compute a Gröbner basis by interval arithmetic. In the author's opinion, this is one of reasons that many researchers still have been studying Gröbner basis with inexact input.

3 Approximate Gröbner Basis with Inexact Input

As in the previous section, unfortunately, treating inexact parts of coefficients as parameters does not give us any reasonable (w.r.t. computation time and number

of segments) answer to the second kind of problem. In this section, we review another approach by Sasaki and Kako [11]. They tried to define approximate Gröbner basis by the following approximate-zero tests for polynomials appearing in the Buchberger algorithm. We note that they also introduced several numerical techniques to prevent cancellation errors and we briefly review only their concept without their complete settings and definitions.

Definition 3 (Approximate-Zero Test). *Let $p(\mathbf{x})$ be a polynomial appearing in the Buchberger algorithm, and $(s_1(\mathbf{x}), \dots, s_{\tilde{k}}(\mathbf{x}))$ be the syzygy for $p(\mathbf{x})$ satisfying $p(\mathbf{x}) = \sum_{i=1}^{\tilde{k}} s_i(\mathbf{x})\tilde{f}_i(\mathbf{x})$. If $\|p\| < \varepsilon \times \max\{\|s_1\tilde{f}_1\|, \dots, \|s_{\tilde{k}}\tilde{f}_{\tilde{k}}\|\}$ where $\|p\|$ denote the infinity norm of $p(\mathbf{x})$, then we say $p(\mathbf{x})$ is approximately zero at tolerance ε , and we denote this as $p(\mathbf{x}) \equiv 0$ (tol ε). \triangleleft*

Definition 4 (Practical Approximate-Zero Test). *Let $p(\mathbf{x})$ be a polynomial appearing in the Buchberger algorithm, and (p_1, \dots, p_m) be all the non-zero coefficients tuple of $p(\mathbf{x})$. If $\max\{|p_1|, \dots, |p_m|\} < \varepsilon$, then we say $p(\mathbf{x})$ is practically approximate-zero at tolerance ε , and we denote this as $p(\mathbf{x}) \equiv 0$ (tol ε). \triangleleft*

With one of the above definitions (computation of syzygies is time-consuming, so they decided to use the second one in practice), they define the following approximate Gröbner basis.

Definition 5 (Approximate Gröbner Basis). *Let ε be a small positive number, and $G = \{g_1, \dots, g_r\}$ be a polynomial set. We call G an approximate Gröbner basis of tolerance ε , if we have $\overline{S(g_i, g_j)}^G \equiv 0$ (tol ε) ($\forall i \neq j$) where $S(g_i, g_j)$ and \overline{p}^G denote the S -polynomial of g_i and g_j and the normal form of p by G , respectively. \triangleleft*

The above definition can be considered as a numerical version of comprehensive Gröbner system with a single parameter by Weispfenning [17], using much reasonably relaxed bounds instead of exact interval arithmetic. In the Buchberger algorithm, head terms of polynomials appearing in the procedure are critically important hence most of known results have to take care of approximate zero tests by exact interval arithmetic, parametric representation or the above way for example. In the rest of the paper, we consider the second kind of problem as a problem in numerical linear algebra instead of trying to extend the Buchberger algorithm directly.

4 Gröbner Basis for Inexact Input as Linear Space

We note again that the first and second kinds of problem are fundamentally different. For the first kind, there exists the answer which is a Gröbner basis of the ideal I generated by F and can be computable by exact arithmetic. On the other hand, for the second one, there exist so many possible answers since F is not known in practice and the given polynomials of \tilde{F} have a priori errors and we can absolutely not be able to know that they should be. Moreover, for the given

\tilde{F} and the unknown F , it may happen that $p(\mathbf{x}) \in \text{ideal}(G)$ and $p(\mathbf{x}) \notin \text{ideal}(F)$ even if we can compute a Gröbner basis G for $\text{ideal}(\tilde{F})$ by some method, where $\text{ideal}(S)$ denotes the ideal generated by the elements of a set S . Because such a Gröbner basis is only a candidate for possible so many Gröbner bases for unknown F . It also be possible that they include $\{1\}$. Any resolution for the second kind of problem must guarantee that $p(\mathbf{x}) \in \text{ideal}(G)$ and $p(\mathbf{x}) \in \text{ideal}(\tilde{F})$ are equivalent with or without some conditions since what is the most reliable is not G but the given \tilde{F} (this is the only reliable information) which does not have any posteriori error. In the below, we give a resolution from this point of view.

4.1 Gröbner Basis as Linear Space

Some researchers studied computing a Gröbner basis by reduced row echelon form ([11], [12]) though there are no concrete algorithms described. However, this is not efficient since we have to operate with large matrices. Using matrix operations partially like F4 and F5 ([13], [14], [2]) may be the best choice if we want to decrease the computation time. We note that the matrix constructed in the F4 algorithm is essentially the same as in this paper and is more compact and well considered. On the other hand, for the second kind of problem, it may be useful since we can use so many results from numerical linear algebra for the situation where we must inevitably operate with a priori errors. Hence we summarize an algorithm for computing Gröbner basis with exact input by reduced row echelon form in this subsection. We note that we use the following definition though there are several equivalents (see [15] or other text books).

Definition 6 (Gröbner Basis). $G = \{g_1, \dots, g_r\} \subseteq I \setminus \{0\}$ is a Gröbner basis for I w.r.t. a fixed term order \succ if for any $f \in I \setminus \{0\}$, there exists $g_i \in G$ such that $\text{ht}(g_i) \geq \text{ht}(f)$ where $\text{ht}(p)$ denotes the head term of $p(\mathbf{x}) \in \mathbf{C}[\mathbf{x}]$ w.r.t. \succ . \triangleleft

We consider the linear map $\phi_{\mathcal{T}} : \mathbf{C}[\mathbf{x}]_{\mathcal{T}} \rightarrow \mathbf{C}^m$ such that $\phi_{\mathcal{T}}(t_i) = \bar{e}_i$ where $\mathbf{C}[\mathbf{x}]_{\mathcal{T}}$ is the submodule of $\mathbf{C}[\mathbf{x}]$ generated by an ordered set (the most left element is the highest) of terms $\mathcal{T} = \{t_1, \dots, t_m\}_{\succ}$ and $\bar{e}_i (i = 1, \dots, m)$ denotes the canonical basis of \mathbf{C}^m . The coefficient vector \bar{p} of $p(\mathbf{x}) \in \mathbf{C}[\mathbf{x}]$ is defined to be satisfying $\bar{p} = \phi_{\mathcal{T}}(p)$ and $p(\mathbf{x}) = \phi_{\mathcal{T}}^{-1}(\bar{p})$. With a fixed \mathcal{T} , we consider the following subset $F_{\mathcal{T}}$ of I .

$$F_{\mathcal{T}} = \left\{ \sum_{i=1}^k s_i(\mathbf{x})f_i(\mathbf{x}) \mid s_i(\mathbf{x})f_i(\mathbf{x}) \in \mathbf{C}[\mathbf{x}]_{\mathcal{T}}, s_i(\mathbf{x}) \in \mathbf{C}[\mathbf{x}] \right\}.$$

The Buchberger algorithm guarantees that $G \subseteq F_{\mathcal{T}}$ if \mathcal{T} has a large enough number of elements. To compute a Gröbner basis for I , we construct the matrix $\mathcal{M}_{\mathcal{T}}(F)$ whose each row vector \bar{p} satisfies $\phi_{\mathcal{T}}^{-1}(\bar{p}) \in \mathcal{P}_{\mathcal{T}}(f)$ for $f(\mathbf{x}) \in F$ where

$$\mathcal{P}_{\mathcal{T}}(p) = \{t_i \times p(\mathbf{x}) \in \mathbf{C}[\mathbf{x}]_{\mathcal{T}} \mid t_i = \phi_{\mathcal{T}}^{-1}(\bar{e}_i), i = 1, \dots, m\}.$$

By this definition, $F_{\mathcal{T}}$ and the linear space $\mathcal{V}_{\mathcal{T}}$ generated by the row vectors of $\mathcal{M}_{\mathcal{T}}(F)$ are isomorphic.

We note that a matrix is said to be in reduced row echelon form if it satisfies the following four conditions.

1. All nonzero rows appear above zero rows.
2. Each leading element of a row is in a column to the right of the leading element of the row above it.
3. The leading element in any nonzero row is 1.
4. Every leading element is the only nonzero element in its column.

Lemma 1. *Let $\overline{\mathcal{M}_{\mathcal{T}}(F)}$ be the reduced row echelon form of $\mathcal{M}_{\mathcal{T}}(F)$. If $g_i(\mathbf{x}) \in F_{\mathcal{T}}$ for a fixed $i \in \{1, \dots, r\}$, $\overline{\mathcal{M}_{\mathcal{T}}(F)}$ has a row vector \vec{p} satisfying $\text{ht}(g_i) = \text{ht}(\phi_{\mathcal{T}}^{-1}(\vec{p}))$. \triangleleft*

Proof. Since the linear map $\phi_{\mathcal{T}}$ is defined by the ordered set \mathcal{T} , each leading element of a row vector \vec{p} of $\overline{\mathcal{M}_{\mathcal{T}}(F)}$ is corresponding to $\text{ht}(\phi_{\mathcal{T}}^{-1}(\vec{p}))$. The lemma follows from the facts that $F_{\mathcal{T}}$ and $\mathcal{V}_{\mathcal{T}}$ are isomorphic and all the leading entries of nonzero rows are disjoint since $\overline{\mathcal{M}_{\mathcal{T}}(F)}$ is in the reduced row echelon form. \square

Lemma 2. *Let $\overline{\mathcal{M}_{\mathcal{T}}(F)}$ be the reduced row echelon form of $\mathcal{M}_{\mathcal{T}}(F)$. If \mathcal{T} has a large enough number of elements, the following $G_{\mathcal{T}}$ is a Gröbner basis for I .*

$$G_{\mathcal{T}} = \left\{ \phi_{\mathcal{T}}^{-1}(\vec{p}) \mid \vec{p} \text{ is a row vector of } \overline{\mathcal{M}_{\mathcal{T}}(F)} \right\}. \quad \triangleleft$$

Proof. The Buchberger algorithm guarantees that $G \subseteq F_{\mathcal{T}}$ if \mathcal{T} has a large enough number of elements. Therefore, $G_{\mathcal{T}}$ satisfies the condition of Definition 6 since we have $g_i(\mathbf{x}) \in G_{\mathcal{T}}$, $i = \{1, \dots, r\}$ by Lemma 1. \square

The above lemmas lead us to the following algorithm directly.

Algorithm 1. (Gröbner Basis by Row Echelon Form)

Input: a term order \succ and a set F of polynomials,

$$F = \{f_1(\mathbf{x}), \dots, f_k(\mathbf{x})\} \subset \mathbf{C}[\mathbf{x}].$$

Output: a Gröbner basis G for the ideal generated by F ,

$$G = \{g_1(\mathbf{x}), \dots, g_r(\mathbf{x})\} \subset \mathbf{C}[\mathbf{x}].$$

1. $d \leftarrow \max_{i=1, \dots, k} \text{tdeg}(f_i)$ (the total degree of $f_i(\mathbf{x})$).
2. $\mathcal{T} \leftarrow$ the ordered set of the terms of total degrees $\leq d$.
3. $\overline{\mathcal{M}_{\mathcal{T}}(F)} \leftarrow$ the reduced row echelon form of $\mathcal{M}_{\mathcal{T}}(F)$.
4. $G_{\mathcal{T}} \leftarrow \left\{ \phi_{\mathcal{T}}^{-1}(\vec{p}) \mid \vec{p} \text{ is a row vector of } \overline{\mathcal{M}_{\mathcal{T}}(F)} \right\}$.
5. $G \leftarrow G_{\mathcal{T}} \setminus \{g \in G_{\mathcal{T}} \mid \exists h \in G_{\mathcal{T}} \setminus \{g\} \text{ s.t. } \text{ht}(h) | \text{ht}(g)\}$.
6. Outputs G if the following conditions satisfied:

- 6-1. $\forall f \in F, \overline{f}^G = 0,$
- 6-2. $\forall g_i, g_j \in G, \overline{S(g_i, g_j)}^G = 0,$

otherwise $d \leftarrow d + 1$ and goto Step 2. \triangleleft

Algorithm [1](#) is not optimized. For example, we should optimize the algorithm as follows. In Step 1, it is better that we start with a larger d (e.g. $\max_{i=1, \dots, k} \text{tdeg}(f_i) + 1$ or a large enough d such that all the S-polynomials of F can be calculated in $\mathbf{C}[x]_{\mathcal{T}}$). Moreover, we can use the rectangular degree (bounding each variable separately and also called the multi degree) instead of the total degree. In Step 6, it is better that we increment d by Δ_d such that $S(g_i, g_j)$ can be calculated in $F_{\mathcal{T}}$ for any pair of elements of G and \mathcal{T} with $d \leftarrow d + \Delta_d$.

Lemma 3. *Algorithm [7](#) computes the reduced Gröbner basis for the ideal generated by the given polynomial set F .* ◁

Proof. The condition 6-1 guarantees that the ideals generated by F and G are the same. Hence, if \mathcal{T} has a large enough number of elements, Algorithm [1](#) outputs a Gröbner basis for the ideal generated by F since the condition 6-2 means that G is a Gröbner basis for the ideal generated by G . Step 5 deletes verbose polynomials by Definition [6](#) hence G is a minimal Gröbner basis. The lemma follows from the fact that $\overline{\mathcal{M}_{\mathcal{T}}(F)}$ is in the reduced row echelon form so that all the polynomials corresponding to row vectors are already reduced by other rows (polynomials). In this algorithm, we use total degree bounds for \mathcal{T} hence \mathcal{T} must have a large enough number of elements in finite steps. ◻

Example 1. We compute the reduced Gröbner basis w.r.t. the graded lexicographic order for the ideal generated by the following polynomials. We note that we show only very simple example since it is difficult to show the whole matrices for nontrivial cases.

$$F = \{2x + 3y, xy - 2\}.$$

In this case, we construct the following matrix $\mathcal{M}_{\mathcal{T}}(F)$ with $d = 3$ and compute its reduced row echelon form $\overline{\mathcal{M}_{\mathcal{T}}(F)}$.

$$\mathcal{M}_{\mathcal{T}}(F) = \begin{pmatrix} 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -2 \end{pmatrix}, \quad \overline{\mathcal{M}_{\mathcal{T}}(F)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{9}{2} & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{4}{3} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{4}{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{3}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Hence, we have the following candidate $G_{\mathcal{T}}$ for a Gröbner basis.

$$\left\{ x^3 - \frac{9y}{2}, yx^2 + 3y, xy^2 - 2y, y^3 + \frac{4y}{3}, x^2 + 3, xy - 2, y^2 + \frac{4}{3}, x + \frac{3y}{2} \right\}.$$

We delete all the verbose elements and test the conditions in Step 6. Since they pass the conditions, we obtain the following reduced Gröbner basis.

$$G = \left\{ x + \frac{3y}{2}, y^2 + \frac{4}{3} \right\}.$$

◁

4.2 Definition of Numerical Gröbner Basis as Linear Space

Let $\mathcal{M}_{\mathcal{T}}(\tilde{F}, p)$ be the matrix whose row vectors are of $\mathcal{M}_{\mathcal{T}}(\tilde{F})$ and $\phi_{\mathcal{T}}(p)$ of a polynomial $p(\mathbf{x})$. We denote the numerical rank of matrix M by $\text{rank}_{\varepsilon}(M)$ which satisfies

$$\text{rank}_{\varepsilon}(M) = \min_{\|M - M'\|_2 \leq \varepsilon} \text{rank}(M')$$

where $\text{rank}(M)$ denotes the conventional matrix rank of M . We note that for any $\kappa < \text{rank}(M)$, we have

$$\min_{\text{rank}(M') = \kappa} \|M - M'\|_2 = \sigma_{\kappa+1}$$

where σ_i denotes the i -th largest singular value of M .

The difference of the ideal membership of $p(\mathbf{x})$, between $\text{ideal}(G) \supseteq \tilde{F}$ and $\text{ideal}(F)$ may increase with increasing the total degree or the number of terms of $p(\mathbf{x})$. Hence, we consider the equivalence of $\text{ideal}(G)$ and $\text{ideal}(\tilde{F})$ by limiting the total degree or the number of terms that must be the lowest value satisfying $G \subset \tilde{F}_{\mathcal{T}}$ since we wish to keep the relations between G and \tilde{F} . We note again that \tilde{F} is only reliable since F is not known.

Definition 7 (Numerical Membership). *For a polynomial $p(\mathbf{x})$, a polynomial set \tilde{F} and an ordered set of terms \mathcal{T} , we say that $p(\mathbf{x})$ is numerically a member of $\text{ideal}(\tilde{F})$ w.r.t. \mathcal{T} and the tolerance ε if $\text{rank}(\mathcal{M}_{\mathcal{T}}(\tilde{F})) = \text{rank}_{\varepsilon}(\mathcal{M}_{\mathcal{T}}(\tilde{F}, p))$. We denote this by $p(\mathbf{x}) \in_{\mathcal{T}, \varepsilon} \text{ideal}(\tilde{F})$. ◁*

By this definition, we say $\text{ideal}(\tilde{F})$ and $\text{ideal}(G)$ are numerically equivalent if and only if $\forall f(\mathbf{x}) \in \tilde{F}, f(\mathbf{x}) \in_{\mathcal{T}, \varepsilon} \text{ideal}(G)$ and $\forall g(\mathbf{x}) \in G, g(\mathbf{x}) \in_{\mathcal{T}, \varepsilon} \text{ideal}(\tilde{F})$. One may think that with this definition some strange situations can happen. For example, it is possible that every polynomials numerically belong to an ideal or that $s_1 f_1 + s_2 f_2$ does not numerically belong to an ideal even if f_1 and f_2 numerically belong to it. This is correct and inevitable for the second kind of problem. \tilde{F} are just one of possible sets for F so we cannot ignore the extreme case: $1 \in \text{ideal}(F)$. Moreover, even if we use exact arithmetic as in Section 2, after any computation (e.g. $s_1 f_1 + s_2 f_2$), the difference from F usually becomes larger hence some strange situations may happen.

The above definition cannot be used for testing $\overline{S(g_i, g_j)}^G = 0$ ($g_i, g_j \in G$) since it usually happens that $S(g_i, g_j) \in_{\mathcal{T}, \varepsilon} \text{ideal}(G)$, depending on \mathcal{T} . We suppose $g_j(\mathbf{x}) \succ g_i(\mathbf{x})$ ($j < i$) and construct the matrix $\mathcal{R}_{\mathcal{T}}(G)$ whose each row vector \vec{p} satisfies $\phi_{\mathcal{T}}^{-1}(\vec{p}) \in \mathcal{P}_{\mathcal{T}}(g_i)$ for $g_i(\mathbf{x}) \in G$ where

$$\mathcal{P}_{\mathcal{T}}(g_i) = \{t_i \times g_i \in \mathbf{C}[\mathbf{x}]_{\mathcal{T}} \mid t_i = \phi_{\mathcal{T}}^{-1}(\vec{e}_i), i = 1, \dots, m, \\ \exists g \in \mathcal{P}_{\mathcal{T}}(g_j) (j < i), \text{ht}(g) = \text{ht}(t_i \times g_i)\}.$$

Similar to $\mathcal{M}_{\mathcal{T}}(\tilde{F}, p)$, $\mathcal{R}_{\mathcal{T}}(G, p)$ is defined as the matrix whose row vectors are the vectors of $\mathcal{R}_{\mathcal{T}}(G)$ and $\phi_{\mathcal{T}}(p)$ of a polynomial $p(\mathbf{x})$.

Definition 8 (Numerical S-Polynomial Check). For polynomials $g_i(\mathbf{x})$ and $g_j(\mathbf{x})$ of a set G and an ordered set of terms \mathcal{T} , we say that the S -polynomial $S(g_i, g_j)$ is numerically reduced to 0 by G w.r.t. \mathcal{T} and the tolerance $\varepsilon \in \mathbf{R}_{\geq 0}$ if $\text{rank}(\mathcal{R}_{\mathcal{T}}(G)) = \text{rank}_{\varepsilon}(\mathcal{R}_{\mathcal{T}}(G, S(g_i, g_j)))$. We denote it by $\overline{S(g_i, g_j)}^G =_{\mathcal{T}, \varepsilon} 0$. \triangleleft

Definition 9 (Numerical Gröbner Basis). We say that $G = \{g_1, \dots, g_r\}$ is a numerical Gröbner basis for $\text{ideal}(\tilde{F})$ w.r.t. a fixed term order \succ and a tolerance $\varepsilon \in \mathbf{R}_{\geq 0}$ if the following conditions are satisfied.

1. $\forall i, j \in \{1, \dots, r\}, \text{lcm}(\text{ht}(g_i), \text{ht}(g_j)) \in \mathcal{T}$,

2. $\forall i, j \in \{1, \dots, r\}, \overline{S(g_i, g_j)}^G =_{\mathcal{T}, \varepsilon} 0$

where \mathcal{T} is an ordered set of terms such that $\text{ideal}(\tilde{F})$ and $\text{ideal}(G)$ are numerically equivalent. In addition, minimal and reduced Gröbner basis are also defined in the ordinary way. \triangleleft

We note that the above definition is compatible with the conventional Gröbner basis since they are the same if $\varepsilon = 0$. Moreover, any conventional Gröbner basis is always a numerical Gröbner basis w.r.t. any tolerance. One may think that this definition for the second kind of problem is not well-posed which is the notion introduced by Hadamard and should have three properties: a solution exists, is unique, and continuously depends on the data. Analyzing the definition from this point of view is postponed for future work.

4.3 How to Compute Numerical Gröbner Basis

Computing a numerical Gröbner basis defined in the previous subsection is not easy. In this subsection, we give a naive method using the reduced row echelon form. Though Algorithm 1 uses only the reduced row echelon form, for the numerical case, we separate it into the forward Gaussian elimination and back-substitution. Let $\mathcal{U}_{\mathcal{T}}(\tilde{F})$ be the upper triangular matrix by the forward Gaussian elimination with partial pivoting, using an unitary transformation (i.g. givens rotation), of $\mathcal{M}_{\mathcal{T}}(\tilde{F})$, and $\mathcal{U}_{\mathcal{T}, \varepsilon}(\tilde{F})$ be the same matrix but neglecting elements and rows that are smaller than the given tolerance ε in absolute value and 2-norm, respectively.

Algorithm 2. (Numerical Gröbner Basis)

Input: a tolerance $\varepsilon \ll 1$, a term order \succ and a set \tilde{F} ,

$$\tilde{F} = \{f_1(\mathbf{x}), \dots, f_k(\mathbf{x})\} \subset \mathbf{C}[\mathbf{x}].$$

Output: a numerical Gröbner basis G for $\text{ideal}(\tilde{F})$,

$$G = \{g_1(\mathbf{x}), \dots, g_r(\mathbf{x})\} \subset \mathbf{C}[\mathbf{x}] \text{ or "failed"}.$$

1. $d \leftarrow \max_{i=1, \dots, n} \text{tdeg}(f_i)$ and $e \leftarrow 1$.
2. $\mathcal{T} \leftarrow$ the ordered set of the terms of total degrees $\leq d$.
3. $\mathcal{U}_{\mathcal{T}, \varepsilon}(\tilde{F}) \leftarrow$ the upper triangular matrix by the forward Gaussian elimination with partial pivoting, using an unitary transformation of $\mathcal{M}_{\mathcal{T}}(\tilde{F})$.
4. $\mathcal{U}_{\mathcal{T}, \varepsilon}(\tilde{F}) \leftarrow$ the reduced row echelon form of $\mathcal{U}_{\mathcal{T}, \varepsilon}(\tilde{F})$
by back-substitution without scaling pivots to one.
5. $G_{\mathcal{T}} \leftarrow \left\{ \phi_{\mathcal{T}}^{-1}(\vec{p}) \mid \vec{p} \text{ is a row of } \mathcal{U}_{\mathcal{T}, \varepsilon}(\tilde{F}), \|\vec{p}\|_2 > \varepsilon \right\}$.

- $\overline{G_{\mathcal{T}}} \leftarrow \left\{ \phi_{\mathcal{T}}^{-1}(\vec{p}) \mid \vec{p} \text{ is a row of } \overline{\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F})}, \|\vec{p}\|_2 > \varepsilon \right\}.$
- 6. $\tilde{G} \leftarrow G_{\mathcal{T}} \setminus \{g \in G_{\mathcal{T}} \mid \exists h \in G_{\mathcal{T}} \setminus \{g\} \text{ s.t. } \text{ht}(h) \mid \text{ht}(g)\}.$
- $G \leftarrow \overline{G_{\mathcal{T}}} \setminus \{g \in \overline{G_{\mathcal{T}}} \mid \exists h \in \overline{G_{\mathcal{T}}} \setminus \{g\} \text{ s.t. } \text{ht}(h) \mid \text{ht}(g)\}.$
- 7. Outputs G or \tilde{G} whichever satisfies the conditions:
 - 7-1. $\forall g_i, g_j \in G, \text{lcm}(\text{ht}(g_i), \text{ht}(g_j)) \in \mathcal{T},$
 - 7-2. $\forall f \in \tilde{F}, f(\mathbf{x}) \in_{\mathcal{T},\varepsilon} \text{ideal}(G),$
 - 7-3. $\forall g_i, g_j \in G, \overline{S(g_i, g_j)}^G =_{\mathcal{T},\varepsilon} 0.$
- 8. Outputs “failed” if $3^e \varepsilon \geq 1.$
- 9. $d \leftarrow d + 1, e \leftarrow e + 1$ and goto Step 2. ◁

Lemma 4. *Throughout Algorithm 2, we have*

$$\forall g \in G_{\mathcal{T}} (\supseteq \tilde{G}), g(\mathbf{x}) \in_{\mathcal{T},\delta} \text{ideal}(\tilde{F})$$

where $\delta = \|\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}) - \mathcal{U}_{\mathcal{T}}(\tilde{F})\|.$ ◁

Proof. Let $\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}, g)$ be the matrix whose row vectors are of $\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F})$ and $\phi_{\mathcal{T}}(g)$, and $\mathcal{U}_{\mathcal{T}}(\tilde{F}, g)$ be the matrix whose row vectors are of $\mathcal{U}_{\mathcal{T}}(\tilde{F})$ and $\phi_{\mathcal{T}}(g)$. By the assumption of the lemma and $\text{rank}(\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}, g)) = \text{rank}(\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}))$, we have $\|\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}, g) - \mathcal{U}_{\mathcal{T}}(\tilde{F}, g)\|_2 \leq \delta$. Since $\mathcal{U}_{\mathcal{T}}(\tilde{F})$ is calculated by only unitary transformations, we have $\mathcal{U}_{\mathcal{T}}(\tilde{F}) = U\mathcal{M}_{\mathcal{T}}(\tilde{F})$ where U denotes the product of such transformations. Let U' be the following unitary matrix satisfying $\mathcal{U}_{\mathcal{T}}(\tilde{F}, g) = U'\mathcal{M}_{\mathcal{T}}(\tilde{F}, g)$.

$$U' = \begin{pmatrix} & & 0 \\ & U & \vdots \\ & & 0 \\ 0 \cdots 0 & & 1 \end{pmatrix}.$$

The lemma follows from the facts that all the singular values of $\mathcal{M}_{\mathcal{T}}(\tilde{F}, g)$ and $U'\mathcal{M}_{\mathcal{T}}(\tilde{F}, g)$ are the same since U' is unitary. ◻

Lemma 5. *Throughout Algorithm 2, we have*

$$\forall g \in \overline{G_{\mathcal{T}}} (\supseteq G), g(\mathbf{x}) \in_{\mathcal{T},\delta} \text{ideal}(\tilde{F})$$

where $\delta = \|\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}) - \mathcal{U}_{\mathcal{T}}(\tilde{F})\|.$ ◁

Proof. Since for any row vector \vec{p} of $\overline{\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F})}$, \vec{p} is a linear combination of row vectors of $\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F})$, we have $\text{rank}(\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}, g)) = \text{rank}(\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}))$. The lemma is proved by the same way in the above proof. ◻

Lemma 6. *Throughout Algorithm 2, we have*

$$\forall f \in \tilde{F}_{\mathcal{T}}, f(\mathbf{x}) \in_{\mathcal{T},\delta} \text{ideal}(G_{\mathcal{T}})$$

where $\delta = \|\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}) - \mathcal{U}_{\mathcal{T}}(\tilde{F})\|.$ ◁

Proof. The lemma follows from the fact $\|\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}, f) - \mathcal{U}_{\mathcal{T}}(\tilde{F}, f)\|_2 \leq \delta$ as in the above proves. \square

Unfortunately, the above lemmas do not guarantee that Algorithm 2 always terminates with a numerical Gröbner basis. However, they suggest $3^e\varepsilon \geq 1$ in Step 8 as follows. One of the reasons that Algorithm 2 can fail to terminate with a numerical Gröbner basis is $\exists g \in \bar{G}, t \in \mathcal{T}, tg \notin_{\mathcal{T},\varepsilon} G_{\mathcal{T}}$. For a proper superset \mathcal{T}' of \mathcal{T} , by the above lemmas, we have

$$\begin{aligned} & \|\mathcal{U}_{\mathcal{T}',\varepsilon}(\tilde{F}, tg) - \mathcal{U}_{\mathcal{T}',\varepsilon}(\tilde{F})\|_2 \\ &= \|\mathcal{U}_{\mathcal{T}',\varepsilon}(\tilde{F}, tg) - \mathcal{U}_{\mathcal{T}'}(\tilde{F}, tg) + \mathcal{U}_{\mathcal{T}'}(\tilde{F}, tg) - \mathcal{U}_{\mathcal{T}'}(\tilde{F}) + \mathcal{U}_{\mathcal{T}'}(\tilde{F}) - \mathcal{U}_{\mathcal{T}',\varepsilon}(\tilde{F})\|_2 \\ &\leq \|\mathcal{U}_{\mathcal{T}',\varepsilon}(\tilde{F}, tg) - \mathcal{U}_{\mathcal{T}'}(\tilde{F}, tg)\|_2 + \|\mathcal{U}_{\mathcal{T}'}(\tilde{F}, tg) - \mathcal{U}_{\mathcal{T}'}(\tilde{F})\|_2 \\ &\hspace{15em} + \|\mathcal{U}_{\mathcal{T}'}(\tilde{F}) - \mathcal{U}_{\mathcal{T}',\varepsilon}(\tilde{F})\|_2 \\ &\leq 3\delta' \end{aligned}$$

where $\delta' = \|\mathcal{U}_{\mathcal{T}',\varepsilon}(\tilde{F}) - \mathcal{U}_{\mathcal{T}'}(\tilde{F})\|$. This means that the distance between \bar{G} and $G_{\mathcal{T}}$ increases by a factor of 3 in the worst case, even if we decrease δ and δ' such that $\delta, \delta' \approx \varepsilon$.

In our preliminary implementation, due to accumulating numerical errors, we use the following $G_{\mathcal{T}}$ and $\overline{G_{\mathcal{T}}}$ instead of the above.

$$\begin{aligned} G_{\mathcal{T}} &\leftarrow \left\{ \phi_{\mathcal{T}}^{-1}(\vec{p}) \mid \vec{p} \text{ is a row of } \mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F}), \|\vec{p}\|_2 > \varepsilon^{1/2} \right\}, \\ \overline{G_{\mathcal{T}}} &\leftarrow \left\{ \phi_{\mathcal{T}}^{-1}(\vec{p}) \mid \vec{p} \text{ is a row of } \overline{\mathcal{U}_{\mathcal{T},\varepsilon}(\tilde{F})}, \|\vec{p}\|_2 > \varepsilon^{1/2} \right\}. \end{aligned}$$

In Step 7, we test G and \bar{G} . However, it is better that we test the all subset of $G_{\mathcal{T}}$ and $\overline{G_{\mathcal{T}}}$ if we do not consider the computing time though we do not implement this. According to our experiments, we could detect a suitable tolerance ε as follows.

$$\varepsilon = 10^{(\log_{10} \sigma_k + \log_{10} \sigma_{k+1})/2} \tag{4.1}$$

where σ_i denotes the i -th largest nonzero singular value of $\mathcal{M}_{\mathcal{T}}(\tilde{F})$ and k is the largest integer maximizes σ_k/σ_{k+1} . Moreover, in our preliminary implementation, we use matrices $\mathcal{N}_{\mathcal{T}}(\tilde{F})$ and $\mathcal{N}_{\mathcal{T}}(\tilde{F}, p)$ instead of $\mathcal{M}_{\mathcal{T}}(\tilde{F})$ and $\mathcal{M}_{\mathcal{T}}(\tilde{F}, p)$, respectively, whose row vectors are normalized in 2-norm. This normalization is not necessary for our definition, however this makes numerical computations more stable.

Example 2. We compute a numerical Gröbner basis w.r.t. the graded lexicographic order and the tolerance $\varepsilon = 10^{-5}$ for the ideal generated by the following polynomials that are the same polynomials in Example 1 but slightly perturbed.

$$\tilde{F} = \{2.000005x + 3.000001y, 0.999999xy - 2.000003\}.$$

In this case, we construct the matrix $\mathcal{N}_{\mathcal{T}}(\tilde{F})$ with $d = 3$ and compute the reduced row echelon form of $\mathcal{N}_{\mathcal{T}}(\tilde{F})$. In Step 5, we have the following candidate for a

numerical Gröbner basis.

$$\overline{G_{\mathcal{T}}} = \{ 0.554701x^3 - 2.49615y, 0.712525yx^2 + 2.13758y, \\ 0.883413xy^2 - 1.76683y, 0.647575y^3 + 0.863437y, \\ 0.554701x^2 + 1.6641, 0.712525xy - 1.42505, \\ 0.522232y^2 + 0.696312, 0.716116x + 1.07417y \}.$$

We delete all the verbose elements and test the conditions in Step 7. Since they pass the conditions, we obtain the following numerical Gröbner basis that are very similar to the result in Example 1.

$$G = \{ 1.0y^2 + 1.33334, 1.0x + 1.5y \}.$$

For the lexicographic order and $\varepsilon = 10^{-5}$, we start with the rectangular degree bound $d = \{2, 2\}$ and we have the following $\overline{G_{\mathcal{T}}}$.

$$\overline{G_{\mathcal{T}}} = \{ 0.447213x^2y^2 - 1.78886, 0.712525yx^2 + 2.13758y, \\ 0.554701x^2 + 1.6641, 0.68755xy^2 - 1.3751y, \\ 0.712525xy - 1.42505, 0.716116x + 1.07417y, 0.522232y^2 + 0.696312 \}.$$

We delete all the verbose elements and test the conditions in Step 7. Since they pass the conditions, we obtain the following numerical Gröbner basis.

$$G = \{ 1.0x + 1.5y, 1.0y^2 + 1.33334 \}. \tag{4.2}$$

Our method can work for the following polynomials having a small head coefficient w.r.t. the lexicographic order.

$$\tilde{F} = \{ 0.0000001x^2 + 2.000005x + 3.000001y, 0.999999xy - 2.000003 \}.$$

With the tolerance $\varepsilon = 6.95972 \times 10^{-9}$ calculated by (4.1) and the rectangular degree bound $d = \{5, 4\}$, we have the following numerical Gröbner basis. We note that the head term of the first element is smaller than ε during inner calculations hence it is not reduced. Moreover, Algorithm 2 outputs the same as in just above (4.2) if we specify $\varepsilon = 10^{-6}$.

$$\{ 0.000861698y^2 + 1.5y + 1.0x + 0.00114879, 1.0y^2 - 0.0000001y + 1.33334 \}.$$

<

5 Remarks

Our approach uses a huge matrix so that it is not effective if we try to compute a Gröbner basis for polynomials with exact coefficients. However, as noted in the beginning of Section 4, it is natural that we use several tools in numerical linear algebra since we have to handle a priori errors and most of symbolic-numeric algorithms for polynomials also use them from necessity. From this point of view, instead of row echelon form by the Gaussian elimination in Algorithm 2, one can use the QR decomposition or the singular value decomposition (SVD) to improve the algorithm though we've not yet analyzed their effectiveness. We note that for all the example in this paper, we use our preliminary implementation on *Mathematica* 6.

Acknowledgements. The author wishes to thank the many referees (especially the one who knows the previous version of this paper) for their useful suggestions that explicitly indicate the future work on the problem mentioned in this paper.

The preliminary implementation code can be found at the following URL.
<http://wwwmain.h.kobe-u.ac.jp/~nagasaka/research/snap/casc09.nb>

References

1. Sasaki, T., Kako, F.: Computing floating-point gröbner bases stably. In: SNC 2007, pp. 180–189. ACM, New York (2007)
2. Kondratyev, A., Stetter, H.J., Winkler, S.: Numerical computation of gröbner bases. In: Proceedings of CASC2004 (Computer Algebra in Scientific Computing), pp. 295–306 (2004)
3. Shirayanagi, K.: An algorithm to compute floating point gröbner bases. In: Proceedings of the Maple summer workshop and symposium on Mathematical computation with Maple V: ideas and applications, pp. 95–106. Birkhauser Boston Inc, Cambridge (1993)
4. Shirayanagi, K.: Floating point gröbner bases. In: Selected papers presented at the international IMACS symposium on Symbolic computation, new trends and developments, pp. 509–528. Elsevier Science Publishers B. V, Amsterdam (1996)
5. Stetter, H.J.: Approximate gröbner bases – an impossible concept? In: Proceedings of SNC 2005 (Symbolic-Numeric Computation), pp. 235–236 (2005)
6. Traverso, C., Zanoni, A.: Numerical stability and stabilization of groebner basis computation. In: ISSAC 2002: Proceedings of the 2002 international symposium on Symbolic and algebraic computation, pp. 262–269. ACM, New York (2002)
7. Weispfenning, V.: Gröbner bases for inexact input data. In: Proceedings of CASC 2003 (Computer Algebra in Scientific Computing), pp. 403–411 (2002)
8. Weispfenning, V.: Comprehensive Gröbner bases. *J. Symbolic Comput.* 14, 1–29 (1992)
9. Weispfenning, V.: Canonical comprehensive Gröbner bases. *J. Symbolic Comput.* 36, 669–683 (2003); International Symposium on Symbolic and Algebraic Computation (ISSAC 2002) (Lille)
10. Nabeshima, K.: A speed-up of the algorithm for computing comprehensive gröbner systems. In: ISSAC 2007: Proceedings of the 2007 international symposium on Symbolic and algebraic computation, pp. 299–306. ACM, New York (2007)
11. Lazard, D.: Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In: van Hulzen, J.A. (ed.) ISSAC 1983 and EUROCAL 1983. LNCS, vol. 162, pp. 146–156. Springer, Heidelberg (1983)
12. Byröd, M., Josephson, K., Åström, K.: Fast optimal three view triangulation. In: Yagi, Y., Kweon, I.S., Kang, S.B., Zha, H. (eds.) Asian Conference on Computer Vision (2007)
13. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases F_4 . *J. Pure Appl. Algebra* 139, 61–88 (1999)
14. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero F_5 . In: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, pp. 75–83. ACM, New York (2002) (electronic)
15. Becker, T., Weispfenning, V.: Gröbner bases. Graduate Texts in Mathematics, vol. 141. Springer, New York (1993); A computational approach to commutative algebra, In cooperation with Heinz Kredel

Modular Algorithms for Computing a Generating Set of the Syzygy Module

Masayuki Noro

Department of Mathematics, Graduate School of Science, Kobe University
noro@math.kobe-u.ac.jp

Abstract. We present two modular algorithms for computing a generating set of the syzygy module of a given sequence of elements in R^l , where R is a polynomial ring or a Weyl algebra over \mathbf{Q} .

1 Introduction

Let R be an n -variate polynomial ring or an n -dimensional Weyl algebra over a field K . For $F = (f_1, \dots, f_k)$, $f_i \in R^l$, its syzygy module $\text{syz}(F) = \{(h_1, \dots, h_k) \mid h_1 f_1 + \dots + h_k f_k = 0\}$ is a (left) submodule of R^k and the computation of a generating set of $\text{syz}(F)$ is an important step for computing various invariants. Algorithms for computing $\text{syz}(F)$ for an arbitrary sequence F are well known and have already been implemented in several computer algebra system, but we found that such implementations do not necessarily compute syzygies efficiently especially over \mathbf{Q} . In this paper we propose improvements of two well-known algorithms for computing syzygies, Algorithm [1](#) and Algorithm [3](#), to which we apply shortcuts by modular computation. Our method is based on the Gröbner trace algorithm originated by C. Traverso [\[14\]](#). In the algorithm, the Buchberger algorithm is executed both over \mathbf{Q} and over a finite field simultaneously and we skip the reduction of an S-polynomial over \mathbf{Q} if it is reduced to zero over the finite field. Then we obtain a probable candidate of a Gröbner basis, but we have to check the candidate to ensure that it is a Gröbner basis of the input ideal. We cannot predict whether this method improves the total efficiency of the Gröbner basis computation of a particular input ideal or not, but we observe that the combination of the trace algorithm and homogenization speeds up the computation in many examples.

When we only want to compute a Gröbner basis, the cost of the check procedure often cancels the gain by skipping the reduction of S-polynomials over \mathbf{Q} . However, if we apply this trace algorithm to Algorithm [1](#), we can make full use of all its parts. First of all the candidate of the reduced Gröbner basis of the input can be computed efficiently. Furthermore, as a by-product of the candidate computation we obtain data for computing a matrix for expressing the Gröbner basis elements as linear combinations of the input polynomials with little extra cost. Then the check procedure is effectively used for producing data necessary for syzygy computation.

We can also apply the trace algorithm to Algorithm 3. If the generating set of an input ideal is a Gröbner basis with respect to a term order, then the check of a Gröbner basis candidate obtained by the trace algorithm is not necessary. That is, if a Gröbner basis candidate is successfully computed, then we can ensure that the candidate is a Gröbner basis of the input ideal. In Algorithm 3, a Gröbner basis computation is executed for an input which is a Gröbner basis with respect to a term order. Therefore we can improve the efficiency of Algorithm 3 by applying the trace algorithm without the check procedure.

2 Algorithms for Computing Syzygies

Let R be an n -variate polynomial ring or an n -dimensional Weyl algebra over a field K . There are two well-known algorithms for computing a generating set of the syzygy module of a sequence F of elements in R^l . In the following algorithm matrices act on column vectors from the left because it is natural for left R -modules.

Algorithm 1 (c.f. [3] chapt. 5, sect. 3)

Input : $F = (f_1, \dots, f_s), f_i \in R^l (i = 1, \dots, s)$
 Output : a generating set of $\text{syz}(F)$
 $G = (g_1, \dots, g_t) \leftarrow$ a Gröbner basis of $\langle F \rangle$ with respect to a term order $<$
 $C \leftarrow$ a (t, s) -matrix s.t. ${}^tG = C \cdot {}^tF$
 $D \leftarrow$ an (s, t) -matrix s.t. ${}^tF = D \cdot {}^tG$
 $S = \{s_1, \dots, s_u\} \leftarrow$ a finite subset of R^t s.t. $\text{syz}(G) = \langle S \rangle$
 $\{r_1, \dots, r_s\} \leftarrow$ rows of $I_s - DC$, where I_s is the unit matrix of size s
 return $\langle s_1C, \dots, s_uC, r_1, \dots, r_s \rangle$

Algorithm 1 is well known and described in many textbooks [1], [3], [5], [6]. Algorithm 1 executes a Gröbner basis computation of $\langle F \rangle$ in R^l , but additional data have to be computed. In principle the matrix C and S can be obtained by keeping track of the coefficients in the reductions of S-polynomials. In practice, however, this adds heavy costs in the execution of the Buchberger algorithm. For general input F , the output is not necessarily a Gröbner basis of $\text{syz}(F)$ and we observe that the computation of a Gröbner basis of $\text{syz}(F)$ is rather hard in general.

Definition 2. Let $<$ be a term order in R . The POT (Position over Term) extension $<_{POT} = (POT, <)$ in R^l of $<$ is a term order defined by $te_i <_{POT} se_j \Leftrightarrow j < i$ or $(i = j \text{ and } t < s)$, where t, s are monomials in R and (e_1, \dots, e_l) is the standard basis of R^l .

Algorithm 3 (c.f. [3] Exercise 15 in chapt. 5, sect. 3)

Input : $F = (f_1, \dots, f_s), f_i \in R^l (i = 1, \dots, s)$, a term order $<$ in R^l
 Output : a Gröbner basis S of $\text{syz}(F)$ with respect to $(POT, <)$,
 a Gröbner basis $G = (g_1, \dots, g_t)$ of $\langle F \rangle$ with respect to $<$ and

```

    a  $(t, s)$ -matrix  $C$  s.t.  ${}^tG = C \cdot {}^tF$ 
 $(e_1, \dots, e_s) \leftarrow$  the standard basis of  $R^s$ 
 $m_i \leftarrow (f_i, e_i) \in R^l \oplus R^s = R^{l+s}$  ( $i = 1, \dots, s$ )
 $\tilde{G} \leftarrow$  a Gröbner basis of  $\langle m_1, \dots, m_s \rangle$  with respect to  $(POT, <)$ 
 $S \leftarrow \{h \in R^s \mid (0, h) \in \tilde{G}\}$ 
 $G \leftarrow \{g \in R^l \mid g \neq 0 \text{ and } (g, h) \in G \text{ for some } h \in R^s\}$ 
 $C \leftarrow$  a  $(t, s)$ -matrix whose  $i$ -th row is  $h_i$  for  $(g_i, h_i) \in \tilde{G}$ 
return  $(S, G, C)$ 

```

We also find Algorithm 34 in several textbooks. In Algorithm 3, we have to execute a Gröbner basis computation in R^{l+s} where s is the number of elements in F , which is harder than that of $\langle F \rangle$ in general. However, all the necessary results can be obtained from the Gröbner basis, and it is important that we have a Gröbner basis of $\text{syz}(F)$ automatically.

3 Gröbner Trace Algorithms

In this section we recall the Gröbner trace algorithm and a theorem 11 which ensures that a Gröbner basis candidate is indeed a Gröbner basis without executing check procedures. For a prime $p \in \mathbf{Z}$, \mathbf{F}_p denotes the finite field of order p . We set $\mathbf{Z}_{(p)} = \{a/b \mid a, b \in \mathbf{Z}, p \nmid b\}$. We define $\phi_p : \mathbf{Z}_{(p)} \rightarrow \mathbf{F}_p$ by $\phi_p(a/b) = (a \bmod p)/(b \bmod p)$.

Notation 4

$\text{LT}(f)$: the monic leading monomial of f
 $\text{LC}(f)$: the leading coefficient of f
 $\text{Spoly}(f, g)$: the S-polynomial of f, g
 $\text{NF}_{G, <}(f)$: a normal form of f with respect to $(G, <)$

Definition 5. Let F be a subset of R^l and $<$ a term order in R^l . A prime p is permissible for $(F, <)$ if for each $f \in F$, $f \in \mathbf{Z}_{(p)}[x_1, \dots, x_n]$ and $\phi_p(\text{LC}(f)) \neq 0$,

The following algorithm computes a Gröbner basis candidate of a submodule $\langle F \rangle$.

Algorithm 6

```

GBCandidate( $F, <, p$ )
Input :  $F \subset \mathbf{Z}[x_1, \dots, x_n]^l$ , a term order  $<$  and a prime  $p$  s.t.  $p$  is permissible
        for  $(F, <)$ 
Output :  $(\text{Status}, G)$ ; if  $\text{Status} = \mathbf{ok}$  then  $G \subset \langle F \rangle$ 
 $D \leftarrow \{\{f, g\} \mid f, g \in F; f \neq g\}$ ;  $G \leftarrow F$ 
while ( $D \neq \emptyset$ ) do
     $C = \{f, g\} \leftarrow$  an element of  $D$ ;  $D \leftarrow D \setminus \{C\}$ 
    if  $\text{NF}_{\phi_p(G), <}(\text{Spoly}(\phi_p(f), \phi_p(g))) \neq 0$  then

```

¹ This is referred as *Caboara-Traverso's algorithm* in [10]. In some textbooks it is often given as an exercise without any reference.

```

    h ← NFG,<(Spoly(f, g))   (h ∈  $\mathbf{Z}_{\langle p \rangle}[X]$ )
    if h ≠ 0 and  $\phi_p(\text{LC}(h)) \neq 0$  then
        D ← D ∪ {{f, h} | f ∈ G};   G ← G ∪ {h}
    else return (ng, ∅)
    endif
end if
end while
G ← the result of removing the redundancy of G
G ← the result of inter-reduction of G return (ok, G)

```

The output G of this algorithm has the following property:

1. $G \subset \langle F \rangle \cap \mathbf{Z}_{\langle p \rangle}[X]$,
2. p is permissible for $(G, <)$,
3. $\phi_p(G)$ is a Gröbner basis of $\langle \phi_p(F) \rangle$.

Definition 7. For $F \subset R^l$, $G \subset \langle F \rangle$ is said to be a p -compatible Gröbner basis candidate of $\langle F \rangle$ with respect to $<$ if G satisfies the above three conditions.

It is often useful for applying homogenization to suppress intermediate coefficient swells.

Algorithm 8

```

GBCandidateHomo(F, <, p)
Fh ← a homogenization of F
<h ← a homogenization of < s.t. the leading term is preserved under
      homogenization and dehomogenization.
(Status, Gh) ← GBCandidate(Fh, <h, p)
If Status = ng return (ng, ∅)
G ← the dehomogenization of Gh
G ← the result of removing the redundancy of G
return G

```

After obtaining a Gröbner basis candidate G , we perform the following two check procedures. If G passes them, then G is a Gröbner basis of $\langle F \rangle$.

Algorithm 9

```

GBCheck(G, <)
Input : a Gröbner basis candidate G, a term order <
Output : status (ok or ng)
If NFG,<(Spoly(f, g)) = 0 for all f, g ∈ G then return ok else return ng.

```

Algorithm 10

```

MemberCheck(F, G, <)
Input : F ⊂ Rl, a Gröbner basis candidate G of ⟨F⟩, a term order <
Output : status (ok or ng)
If NFG,<(f) = 0 for all f ∈ F then return ok else return ng.

```

We can omit the checks if the input is known to be a Gröbner basis with respect to a term order $<_0$.

Theorem 11. *Suppose that $G_0 \subset \mathbf{Z}[x_1, \dots, x_n]^l$ is a Gröbner basis of $\langle G_0 \rangle \subset R^l$ with respect to a term order $<_0$ and p is permissible for $(G_0, <_0)$. Then the following hold:*

1. $\phi_p(\langle G_0 \rangle \cap \mathbf{Z}[x_1, \dots, x_n]^l) = \langle \phi_p(G_0) \rangle$ and $\phi_p(G_0)$ is a Gröbner basis of $\langle \phi_p(G_0) \rangle$.
2. Let $G \subset \langle G_0 \rangle \cap \mathbf{Z}[x_1, \dots, x_n]^l$ be a p -compatible Gröbner basis candidate of $\langle G_0 \rangle$ with respect to $<$. Then G is a Gröbner basis of $\langle G_0 \rangle$ with respect to $<$.

Proof. 1. As $\langle \phi_p(G_0) \rangle \subset \phi_p(\langle G_0 \rangle \cap \mathbf{Z}[x_1, \dots, x_n]^l)$, it is sufficient to show that for any $f \in \langle G_0 \rangle \cap \mathbf{Z}[x_1, \dots, x_n]^l$, $\phi_p(f)$ is reduced to 0 by $\phi_p(G)$. For any $f \in \langle G_0 \rangle \cap \mathbf{Z}[x_1, \dots, x_n]^l$, f is reduced to 0 by G_0 . This reduction procedure is written as follows:

$$f_0 = f, f_i = f_{i-1} - \alpha_i t_i g_{k_i}, f_m = 0 \text{ for some } m,$$

where $\alpha_i \in \mathbf{Q}$, t_i is a monomial and $g_{k_i} \in G_0$. As p is permissible for $(G_0, <_0)$, the denominator of α_i is not divisible by p for each i . Thus we have $\phi_p(f_i) = \phi_p(f_{i-1}) - \phi_p(\alpha_i) t_i \phi_p(g_{k_i})$ and this recurrence represents a reduction of $\phi_p(f_0)$ by $\phi_p(G_0)$. $\phi_p(f_m) = 0$ implies that there exists $i \leq m$ s.t. $\phi_p(f_{i-1}) \neq 0$ and $\phi_p(f_i) = 0$, which means that $\phi_p(f)$ is reduced to 0 by $\phi_p(G_0)$.

2. We show that every $f \in \langle G_0 \rangle$ is reduced to 0 by G . We may assume that f is G -reduced. If $f \neq 0$ then, by multiplying a rational number, we may assume that $f \neq 0$ is a G -reduced element of $\mathbf{Z}[x_1, \dots, x_n]^l$ and the integer content of f is equal to 1. Then $\phi_p(f) \neq 0$, otherwise the integer content of f would have a factor p . As $f \in \langle G_0 \rangle \cap \mathbf{Z}[x_1, \dots, x_n]^l$, $\phi_p(f) \in \phi_p(\langle G_0 \rangle \cap \mathbf{Z}[x_1, \dots, x_n]^l) = \langle \phi_p(G_0) \rangle = \langle \phi_p(G) \rangle$, $\phi_p(f)$ must be reduced to 0 by $\phi_p(G)$. But f is G -reduced and the permissibility of p for $(G, <)$ implies that the set of leading terms of $\phi_p(G)$ is the same as that of G , thus $\phi_p(f)$ is $\phi_p(G)$ -reduced. This is a contradiction. \square

Remark 12. *The above algorithms and theorems are still correct if we replace a polynomial ring $S[x_1, \dots, x_n]$ in the theorems with a Weyl algebra $S\langle x_1, \dots, x_n, \partial_1, \dots, \partial_n \rangle$ where S is a coefficient ring.*

4 Improvements of Algorithms for Computing Syzygies

In order to apply the trace algorithms to Algorithm [11](#), we modify Algorithm [6](#), [9](#) and [10](#) so that they output enough informations for reproducing the normal form computations, which we call trace informations. The modified algorithms are named *GBCandidateEx*, *GBCheckEx* and *MemberCheckEx* respectively. Let F and G be an input and the output of *GBCandidate* respectively. In *GBCandidateEx* we first collect the trace information for each reduction of S -polynomial over \mathbf{Q} . It is the following list:

$$[l, [1, i, m, 1], [1, j, m', 1], [a_1, k_1, m_1, d_1], \dots, [a_s, k_s, m_s, d_s]], \tag{1}$$

which means that

1. $r \leftarrow mg_i + m'g_j = c\text{Spoly}(g_i, g_j)$, where $a, b \in \mathbf{Z}$, m, m' are monomials, $c \in \mathbf{Q}$ and $r \in \mathbf{Z}[x_1, \dots, x_n]$,
2. $r \leftarrow (a_j r + m_j g_{k_j})/d_j$ ($i = 1, \dots, s$), where $a_j, c_j, d_j \in \mathbf{Z}$, m_j is a monomial, $r \in \mathbf{Z}[x_1, \dots, x_n]$ and the coefficients of r is kept as small as possible, and
3. $g_l \leftarrow r$.

The list **(II)** is the data for computing g_l from g_1, \dots, g_{l-1} . Inter-reduction is also expressed by similar lists and finally we can express each $g \in G$ in terms of F by recursive computations, which gives each rows of the matrix C . In *GBCheckEx*, the set of S-polynomials to check the candidate G being Gröbner basis coincides with the set of S-polynomials whose coefficient vectors generate $\text{syz}(\text{LT}(G))$. Here the coefficient vector of an S-polynomial $ag_i - bg_j$ is $ae_i - be_j$ for the standard bases e_i and e_j . Then these S-polynomials are actually reduced over \mathbf{Q} and we obtain the trace informations similar to **(II)**. In *MemberCheckEx*, polynomials in F are reduced by the candidate G over \mathbf{Q} and we obtain the trace informations for computing the matrix D . We notice that the generating set S of $\text{syz}(G)$ is computed only for the dehomogenized one even if we apply *GBCandidateHomoEx* expecting efficient computation of the candidate G .

Algorithm 13 (Modular trace version of Algorithm **(II))**

Input : $F = (f_1, \dots, f_s)$, $f_i \in \mathbf{Z}[x_1, \dots, x_n]^l$ ($i = 1, \dots, s$)

Output : a generating set of $\text{syz}(F)$

do

restart:

$p \leftarrow$ a new unused prime which is permissible for $(F, <)$

$(\text{Status}, G, \text{Ctrace}) \leftarrow \text{GBCandidateEx}(F, <, p)$

if $\text{Status} = \mathbf{ng}$ goto restart

$(\text{Status}, \text{Strace}) \leftarrow \text{GBCheckEx}(G, <)$

if $\text{Status} = \mathbf{ng}$ goto restart

$(\text{Status}, \text{Dtrace}) \leftarrow \text{MemberCheckEx}(F, G, <)$

if $\text{Status} = \mathbf{ng}$ goto restart

$C \leftarrow$ a matrix s.t. ${}^tG = C \cdot {}^tF$ computed from Ctrace

$D \leftarrow$ a matrix s.t. ${}^tF = D \cdot {}^tG$ computed from Dtrace

$S = \{s_1, \dots, s_u\} \leftarrow$ a generating set of $\text{syz}(G)$ computed from Strace

$\{r_1, \dots, r_s\} \leftarrow$ rows of $I_s - DC$, where I_s is the unit matrix of size s

return $\langle s_1C, \dots, s_uC, r_1, \dots, r_s \rangle$

end do

Next we show how one can apply the trace algorithm and Theorem **(II)** to Algorithm **(3)**. We show that the input of the Gröbner basis computation in Algorithm **(3)** already forms a Gröbner basis with respect to a term order.

Theorem 14. For $f_i \in R^l$ ($i = 1, \dots, s$), we set $m_i = (e_i, f_i) \in R^s \oplus R^l = R^{s+l}$, where (e_1, \dots, e_s) is the standard basis of R^s . Then $M = (m_1, \dots, m_s)$ is a Gröbner basis of $\langle M \rangle$ with respect to any POT order.

Proof. If we execute the Buchberger algorithm for M , there are no S-pairs to be reduced because $LT(m_i) = e_i$ with respect to any POT order. Thus M is a Gröbner basis of $\langle M \rangle$.

Theorem 11 and 14 ensures that the following algorithm returns the same output as Algorithm 3.

Algorithm 15 (Modular trace version of Algorithm 3)

```

Input :  $F = (f_1, \dots, f_s)$ ,  $f_i \in \mathbf{Z}[x_1, \dots, x_n]^l$  ( $i = 1, \dots, s$ ), a term order  $<$  in  $R^l$ 
Output : a Gröbner basis  $S$  of  $\text{syz}(F)$  with respect to  $(POT, <)$ ,
        a Gröbner basis  $G = (g_1, \dots, g_t)$  of  $\langle F \rangle$  with respect to  $<$  and
        a  $(t, s)$ -matrix  $C$  s.t.  ${}^tG = C \cdot {}^tF$ 
 $(e_1, \dots, e_s) \leftarrow$  the standard basis of  $R^s$ 
 $m_i \leftarrow (f_i, e_i) \in R^l \oplus R^s = R^{l+s}$  ( $i = 1, \dots, s$ )
do
  restart:
   $p \leftarrow$  a new unused prime which is permissible for  $((m_1, \dots, m_s), (POT, <))$ 
   $(Status, \tilde{G}) \leftarrow GBCandidate((m_1, \dots, m_s), (POT, <), p)$ 
  if  $Status = \mathbf{ng}$  goto restart
   $S \leftarrow \{h \in R^s \mid (0, h) \in \tilde{G}\}$ 
   $G \leftarrow \{g \in R^l \mid g \neq 0 \text{ and } (g, h) \in \tilde{G} \text{ for some } h \in R^s\}$ 
   $C \leftarrow$  a  $(t, s)$ -matrix whose  $i$ -th row is  $h_i$  for  $(g_i, h_i) \in \tilde{G}$ 
end do
return  $(S, G, C)$ 
    
```

5 Experiments

We implemented Algorithm 13 and 15 in Risa/Asir 12 and made a number of experiments. In Algorithm 13, the matrices C , D and the syzygies S are computed by the user language of Risa/Asir from the trace information given in the outputs of a built-in function `nd_gr_trace`, which executes the Gröbner trace algorithm. In this function Algorithm 6 or 8 is executed with a prime $p \simeq 10^8$. Then the obtained candidate is checked by Algorithm 9 and 10. If the check fails then another candidate is computed with another prime p . The main task in Algorithm 15 is the computation of a Gröbner basis candidate, which is also done by `nd_gr_trace`. Our experiments show that computing the syzygy of a general input is very hard in general. We tried many examples of ideals for which Gröbner basis can be easily computed, but we could compute the syzygies of only a few ones among them. Nevertheless the timings of the successful results will show that the new algorithms efficiently compute syzygies for certain inputs. Timings were measured on a Linux machine with Intel Xeon X5470, 3.33GHz and are given in seconds. Here we refer again to the validity check in the modular algorithms. In Algorithm 13, a prime p is valid if `GBCandidateEx`, `GBCheckEx` and `MemberCheckEx` return `ok`. In Algorithm 15, a prime p is valid if `GBCandidateEx` returns `ok`. The default prime 99981793 was valid for all candidate computations.

5.1 Computation over a Commutative Polynomial Ring

We show timings of Algorithm 13, Algorithm 3, Algorithm 15 in Risa/Asir. SINGULAR 5 provides a function `syz` to compute a generating set of a syzygy module. From the outputs of `syz` and the description in 5 we guess that SINGULAR implements Algorithm 3 and we also show its timing for comparing performances with the term order set to (c, dp) , that is the POT extension of the graded reverse lex order. For Algorithm 13 and 15, we tried both *GBCandidate* and *GBCandidateHomo* for computing a Gröbner basis candidate. (H) in the table indicates that the timing was obtained by *GBCandidateHomo* because it took too much time by the non-homogenized computation. We note that Algorithm 3 and 15 compute a Gröbner basis of the syzygy module, but Algorithm 13 simply outputs its generating set. *Cyclic7*, *Kotsireas*, *Virasoro*, *Cohn3* and

Table 1. Timing data of syzygy computations over polynomial rings

Input	Algorithm 13	Algorithm 3	Algorithm 15	SINGULAR
<i>Cyclic7</i>	> 2h	–	882(H)	> 2h
<i>HCyclic7</i>	658	2727	244	3557
<i>Kotsireas</i>	> 2h	–	1863	> 11h
<i>Virasoro</i>	936	> 2h	685	> 3h
<i>Cohn3</i>	3642(H)	–	267(H)	>1.5h
<i>Fabrice24</i>	5382	*	*	*
<i>ABBA₄</i>	8h	*	> 72h	*

Fabrice24 are taken from 4. *HCyclic7* is the homogenized *Cyclic7*, that is $(Cyclic7 \setminus \{x_1 \cdots x_7 - 1\}) \cup \{x_1 \cdots x_7 - x^7\}$ with the graded reverse lex order s.t. $x > x_1 > \cdots > x_7$. *ABBA₄* is the set of all the entries of $AB - BA$ where $A = (a_{ij})$, $B = (b_{ij})$ are 4×4 matrices. In Table 1 ‘–’ and ‘*’ mean that we have not executed the computation because the computation of Algorithm 15 over \mathbf{Q} , or \mathbf{F}_p respectively indicate that it will take very long time to perform the computation.

Table 1 shows that computation of syzygy over \mathbf{Q} without modular shortcuts is very inefficient or practically impossible. Taking the fact that Algorithm 15 outputs a Gröbner basis of the syzygy module into account, we may say that Algorithm 15 is preferable for computing syzygies. However if we only want to know a generating set of the syzygy module, Algorithm 13 may be chosen because Algorithm 15 could not output a result in two examples because of the difficulty of the computation of the Gröbner basis candidate.

5.2 Computation over Weyl Algebra

We compute the syzygy of A-hypergeometric system $H_A(\beta)$, where A is a $d \times n$ integer matrix and β is a d -dimensional column vector. For the following (A_i, β_i) ($i = 1, \dots, 4$), we generate $H_{A_i}(\beta_i)$ and compute its syzygy. See 13 the for detail of $H_A(\beta)$. We show the results of Algorithm 13, 3 and 15. Macaulay2

§ provides a function `syz` in `Dmodules` package. Although we don't know the algorithm implemented in `Macaulay2`, we also show its results for reference.

$$\begin{aligned}
 A_1 &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 \end{pmatrix}, & \beta_1 &= \begin{pmatrix} 123 \\ 456 \end{pmatrix} \\
 A_2 &= \begin{pmatrix} 1 & 1 & 3 & 2 & 1 & 1 \\ 0 & 2 & 1 & 1 & 3 & 2 \\ 2 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}, & \beta_2 &= \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \\
 A_3 &= \begin{pmatrix} 1 & 2 & 3 & 2 & 1 & 1 \\ 1 & 4 & 3 & 3 & 4 & 5 \\ 4 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}, & \beta_3 &= \begin{pmatrix} 5 \\ 8 \\ 12 \end{pmatrix} \\
 A_4 &= \begin{pmatrix} 1 & 2 & 3 & 2 & 1 & 1 \\ 1 & 4 & 3 & 3 & 4 & 5 \\ 4 & 0 & 0 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 3 & 3 \end{pmatrix}, & \beta_4 &= \begin{pmatrix} 5 \\ 8 \\ 12 \\ 6 \end{pmatrix}
 \end{aligned}$$

We only show $H_{A_4}(\beta_i)$.

$$\begin{aligned}
 H_{A_4}(\beta_i) &= \{x_6\partial_6 + x_5\partial_5 + 2x_4\partial_4 + 3x_3\partial_3 + 2x_2\partial_2 + x_1\partial_1 - 5, \\
 &5x_6\partial_6 + 4x_5\partial_5 + 3x_4\partial_4 + 3x_3\partial_3 + 4x_2\partial_2 + x_1\partial_1 - 8, \\
 &x_5\partial_5 + x_4\partial_4 + 4x_1\partial_1 - 12, \\
 &3x_6\partial_6 + 3x_5\partial_5 + 4x_4\partial_4 + 3x_3\partial_3 + 2x_2\partial_2 + x_1\partial_1 - 6, \\
 &-\partial_3\partial_5 + \partial_2\partial_4, -\partial_1\partial_3\partial_6^4 + \partial_2^2\partial_5^4, -\partial_1\partial_4\partial_6^4 + \partial_2\partial_5^5, \partial_1\partial_4^2\partial_6^4 - \partial_3\partial_5^6\}
 \end{aligned}$$

Table 2. Timing data of syzygy computations of $H_A(\beta)$

Input	Algorithm 13	Algorithm 3	Algorithm 15	Macaulay2
(A_1, β_1)	19	47	2.3	32
(A_2, β_2)	51	741	71	> 1h
(A_3, β_3)	126	334	26	544
(A_4, β_4)	18	21	2.5	88

As in the polynomial case, Table 2 shows that Algorithm 15 is a good choice if one wants to know a Gröbner basis of the syzygy module. From the comparison of the results of Algorithm 3 and 15 we see that the trace algorithm efficiently avoids the difficulty caused by large intermediate coefficients. We may say that Algorithm 15 is efficient even if `Macaulay2` outputs a Gröbner basis of the syzygy module.

5.3 Discussions

If an input sequence F is already a Gröbner basis, then a generating set of the syzygy module is easily obtained from the results of reductions of S-polynomials

and it is a Gröbner basis of the syzygy module with respect to a special order (Schreyer order). Möller et al. [9] proposed an algorithm for computing a Gröbner basis G of an input ideal and a Gröbner basis of the syzygy module of G simultaneously. For computing a free resolution of the module generated by a sequence F , we don't have to compute the syzygy of F . Instead we can replace F with a Gröbner basis and we can apply various efficient algorithms such as Schreyer resolution [5] or La Scala-Stillman's algorithm [7]. However, if F is not a Gröbner basis, our experiments show that it is not an easy task to obtain the generator set of the syzygy of F itself. In such a case the proposed methods in this paper may efficiently give the result.

Another modular method for improving the efficiency of Gröbner basis computations was proposed in [2]. In the algorithm, a Gröbner basis candidate is constructed by Chinese remainder computation or Hensel lifting and the validity of the candidate is checked. If the elements of the resulting Gröbner basis have small coefficients then the candidate computation will be efficient. But the input of the algorithm must be homogeneous and the check contains Gröbner basis check of the homogeneous candidate, which is often a heavy computation. In Algorithm [15] we have to compute the candidates over \mathbf{Q} but the check is not necessary. It will be interesting to compare the two methods for various inputs.

References

1. Adams, W., Loustaunau, P.: An Introduction to Gröbner Bases. Graduate Studies in Mathematics, vol. 3. AMS (1994)
2. Arnold, E.: Modular Algorithms for computing Gröber bases. *J. Symb. Comp.* 35(4), 403–419 (2003)
3. Cox, D., Little, J., O'Shea, D.: Using Algebraic Geometry. In: GTM, vol. 185, Springer, Heidelberg (2005)
4. Examples in the web page of Janet Basis, <http://invo.jinr.ru/examples.phtml>
5. Greuel, G.-M., Pfister, G.: A Singular Introduction to Commutative Algebra. Springer, Heidelberg (2007), <http://www.singular.uni-kl.de/>
6. Kreuzer, M., Robbiano, L.: Computational Commutative Algebra, vol. 1. Springer, Heidelberg (2008)
7. La Scala, R., Stillman, M.: Strategies for computing minimal free resolutions. *J. Symb. Comp.* 26, 409–431 (1998)
8. Macaulay 2 home page, <http://www.math.uiuc.edu/Macaulay2/>
9. Möller, H.M., Mora, T., Traverso, C.: Gröber Bases Computation Using Syzygies. In: Proc. ISSAC 1992, pp. 320–328. ACM Press, New York (1992)
10. Mora, T.: Solving Polynomial Equation Systems II, Macaulay's paradigm and Gröber Technology. In: Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge (2005)
11. Noro, M., Yokoyama, K.: A Modular Method to Compute the Rational Univariate Representation of Zero-Dimensional Ideals. *J. Symb. Comp.* 28(1), 243–263 (1999)
12. Risa/Asir: A computer algebra system, <http://www.math.kobe-u.ac.jp/Asir/asir.html>
13. Saito, M., Sturmfels, B., Takayama, N.: Gröber deformations of hypergeometric differential equations. *Algorithms and Computation in Mathematics* 6 (2000)
14. Traverso, C.: Gröber trace algorithms. In: Gianni, P. (ed.) ISSAC 1988. LNCS, vol. 358, pp. 125–138. Springer, Heidelberg (1989)

A Symbolic Framework for Operations on Linear Boundary Problems

Markus Rosenkranz¹, Georg Regensburger¹,
Loredana Tec², and Bruno Buchberger²

¹ Johann Radon Institute for Computational and Applied Mathematics,
Austrian Academy of Sciences, Altenberger Str. 69, 4040 Linz, Austria

² Research Institute for Symbolic Computation,
Johannes Kepler Universität, 4032 Castle of Hagenberg, Austria

Abstract. We describe a symbolic framework for treating linear boundary problems with a generic implementation in the Theorema system. For ordinary differential equations, the operations implemented include computing Green's operators, composing boundary problems and integro-differential operators, and factoring boundary problems. Based on our factorization approach, we also present some first steps for symbolically computing Green's operators of simple boundary problems for partial differential equations with constant coefficients. After summarizing the theoretical background on abstract boundary problems, we outline an algebraic structure for partial integro-differential operators. Finally, we describe the implementation in Theorema, which relies on functors for building up the computational domains, and we illustrate it with some sample computations including the unbounded wave equation.

Keywords: Linear boundary problem, Green's operator, Integro-Differential Operator, Ordinary Differential Equation, Wave Equation.

1 Introduction

Due to their obvious importance in applications, *boundary problems* play a dominant role in Scientific Computing, but almost exclusively in the numerical segment. It is therefore surprising that they have as yet gained little attention in Symbolic Computation, neither from a theoretical perspective nor in computer algebra systems.

In applications [1, p. 42] one is “concerned not only with solving [the boundary problem] for specific data but also with finding a suitable form for the solution that will exhibit its dependence on the data.” In our work, we focus on linear boundary problems (and will henceforth suppress the attribute “linear”). For us, a boundary problem is thus a differential equation with a symbolic right-hand side, supplemented by suitable boundary conditions. Solving it means to determine its *Green's operator*, namely the integral operator that maps the right-hand side to the solution. For a symbolic approach to boundary problems, one has to develop a constructive algebraic theory of integral operators and an algorithmic framework for manipulating boundary conditions.

Such a development was initiated in [2], leading to a symbolic method for computing Green’s operators of regular two-point boundary problems with constant coefficients [3]. We extended these results to a *differential algebra setting* in [4], where we also developed a *factorization method* applicable to boundary problems for ordinary differential equations (ODEs). A more *abstract view* on boundary problems and a general factorization theory is described in [5], including in particular partial differential equations (PDEs).

In this paper, we describe a prototype *implementation in Theorema* [6], currently based on a raw interface that will be improved in the future. It provides generic algorithms for various operations on boundary problems and integro-differential operators for ODEs (Section 5), exemplified in (Appendix A): computing Green’s operators, composing boundary problems and integro-differential operators, and factoring boundary problems. The computations are realized by a suitable noncommutative Gröbner basis that reflects the essential interactions between certain basic operators. Gröbner bases were introduced by Buchberger in [7]. For an introduction to the theory, we refer to [8], for its noncommutative extension to [9].

Moreover, for *PDEs* we present some first steps for making the abstract setting of [5] algorithmic. We develop an algebraic language for encoding the integro-differential operators appearing as Green’s operators of some simple two-dimensional Dirchlet problems for PDEs with constant coefficients (Section 4). Using our generic factorization approach, this allows to find the Green’s operator of higher-order boundary problems by composing those of its lower-order factors. This idea is exemplified for the unbounded wave equation with a sample computation (Appendix A).

For the broader audience of Scientific Computing, we summarize the necessary *theoretical background* on abstract boundary problems, omitting all technical details and illustrating it for the case of ODEs (Section 2). After explaining the composition and factorization of boundary problems (Section 3), we outline the algebraic structures used for encoding ordinary as well as partial integro-differential operators (Section 4).

For motivating our algebraic setting of boundary problems, we consider first the *simplest two-point boundary problem*. Writing \mathcal{F} for the real or complex vector space $C^\infty[0, 1]$, it reads as follows: Given $f \in \mathcal{F}$, find $u \in \mathcal{F}$ such that

$$\boxed{\begin{matrix} u'' = f, \\ u(0) = u(1) = 0. \end{matrix}} \tag{1}$$

Let $D: \mathcal{F} \rightarrow \mathcal{F}$ denote the usual derivation and L, R the two linear functionals $L: f \mapsto f(0)$ and $R: f \mapsto f(1)$. Note that u is annihilated by any linear combination of these functionals so that problem (1) can be described by $(D^2, [L, R])$, where $[L, R]$ is the subspace generated by L, R in the dual space \mathcal{F}^* .

As a second example, consider the following boundary problem for the *wave equation* on the domain $\Omega = \mathbb{R} \times \mathbb{R}_{\geq 0}$, now writing \mathcal{F} for $C^\infty(\Omega)$: Given $f \in \mathcal{F}$, find $u \in \mathcal{F}$ such that

$$\boxed{\begin{aligned} u_{tt} - u_{xx} &= f, \\ u(x, 0) &= u_t(x, 0) = 0. \end{aligned}} \tag{2}$$

Note that we use the terms “boundary condition/problem” in the general sense of linear conditions. The boundary conditions in (2) can be expressed by the infinite family of linear functionals $\beta_x: u \mapsto u(x, 0)$, $\gamma_x: u \mapsto u_t(x, 0)$ with x ranging over \mathbb{R} . So we can represent the boundary problem again by a pair consisting of the differential operator $D_t^2 - D_x^2$ and the (now infinite dimensional) subspace generated by β_x and γ_x in \mathcal{F}^* .

For ensuring a unique representation of boundary conditions, we take the *orthogonal closure* of this subspace, which we denote by $[\beta_x, \gamma_x]_{x \in \mathbb{R}}$. This is the space of all linear functionals vanishing on the functions annihilated by β_x, γ_x . Every finite dimensional subspace is orthogonally closed, but here, for example, the functionals $u \mapsto \int_0^x u(\eta, 0) d\eta$ and $u \mapsto u_x(x, 0)$ for arbitrary $x \in \mathbb{R}$ are in the orthogonal closure but not in the space generated by β_x and γ_x . We refer to [10] or [5, App. A.1] for details on the orthogonal closure.

Some *notational conventions*. We use the symbol \leq for algebraic substructures. If $T: \mathcal{F} \rightarrow \mathcal{G}$ is a linear map and $\mathcal{B} \leq \mathcal{G}^*$, we write $\mathcal{B} \cdot T$ for the subspace $\{\beta \circ T \mid \beta \in \mathcal{B}\} \leq \mathcal{F}^*$. For a subset $\mathcal{B} \subseteq \mathcal{F}^*$ the so-called *orthogonal* is defined as $\mathcal{B}^\perp = \{u \in \mathcal{F} \mid \beta(u) = 0 \text{ for all } \beta \in \mathcal{B}\}$.

2 An Algebraic Formulation of Boundary Problems

In this section, we give a summary of the algebraic setting for boundary problems exposed in [5], see also there for further details and proofs. We illustrate the definitions and statements for ODEs on a compact interval $[a, b] \subseteq \mathbb{R}$. In this setting, most of the statements can be made algorithmic relative to solving homogeneous linear differential equations (and the operations of integration and differentiation).

A *boundary problem* is given by a pair (T, \mathcal{B}) , where $T: \mathcal{F} \rightarrow \mathcal{G}$ is a surjective linear map between vector spaces \mathcal{F}, \mathcal{G} and $\mathcal{B} \leq \mathcal{F}^*$ is an orthogonally closed subspace of homogeneous boundary conditions. We say that $u \in \mathcal{F}$ is a solution of (T, \mathcal{B}) for a given $f \in \mathcal{G}$ if $Tu = f$ and $u \in \mathcal{B}^\perp$. Note that we have restricted ourselves to homogeneous conditions because the general solution is then obtained by adding a “particular solution” satisfying the inhomogeneous conditions. While for ODEs, this amounts to a simple interpolation problem, the treatment of PDEs is more involved.

In the *ODE setting*, $T = D^n + c_{n-1}D^{n-1} + \dots + c_1D + c_0$ is a monic differential operator of order n with coefficients $c_i \in \mathcal{G}$. For the spaces \mathcal{F}, \mathcal{G} we could for example choose $\mathcal{F} = \mathcal{G} = C^\infty[a, b]$ or $\mathcal{F} = C^n[a, b]$ and $\mathcal{G} = C[a, b]$, as real or complex vector spaces. The differential operator T is surjective since every inhomogeneous linear differential equation has a solution in \mathcal{F} , e.g. given by the formula (3) below. The solution space of the homogeneous equation, $\text{Ker } T$, has dimension n , so we require $\dim \mathcal{B} = n$, and we assume that \mathcal{B} is given by a

basis β_1, \dots, β_n . Then the boundary problem reads as follows: Given $f \in \mathcal{G}$, find $u \in \mathcal{F}$ such that

$$\boxed{\begin{aligned} Tu &= f, \\ \beta_1(u) &= \dots = \beta_n(u) = 0. \end{aligned}}$$

The boundary conditions can in principle be any linear functionals. In particular, they can be point evaluations of derivatives or also more general boundary conditions of the form $\beta(u) = \sum_{i=0}^{n-1} a_i u^{(i)}(a) + b_i u^{(i)}(b) + \int_a^b v(\xi) u(\xi) d\xi$ with $v \in \mathcal{F}$, known in the literature [11] as ‘‘Stieltjes boundary conditions’’. Integral boundary conditions also appear naturally when we factor a boundary problem along a given factorization of the differential operator (Section 3), and they appear in the normal forms of integro-differential operators (Section 4).

A boundary problem (T, \mathcal{B}) is *regular* if for each $f \in \mathcal{G}$ there exists exactly one solution u of (T, \mathcal{B}) . Then we call the linear operator $G: \mathcal{G} \rightarrow \mathcal{F}$ that maps a right-hand side f to its unique solution $u = Gf$ the *Green’s operator* for the boundary problem (T, \mathcal{B}) , and we say that G solves the boundary problem (T, \mathcal{B}) . Since $TGf = f$, we see that the Green’s operator for a regular boundary problem (T, \mathcal{B}) is a right inverse of T , determined by the property $\text{Im } G = \mathcal{B}^\perp$. Therefore we use the notation $G = (T, \mathcal{B})^{-1}$ for the Green’s operator.

Regular boundary problems can be characterized as follows. A boundary problem is regular iff \mathcal{B}^\perp is a complement of $\text{Ker } T$ so that $\mathcal{F} = \text{Ker } T \dot{+} \mathcal{B}^\perp$ as a direct sum. For ODEs we have the following algorithmic regularity test (compare [12, p. 184] for the special case of two-point boundary conditions): A boundary problem (T, \mathcal{B}) for an ODE is regular iff the *evaluation matrix* $B = (\beta_i(u_j))$ is regular, where the β_i and u_j are any basis of respectively \mathcal{B} and $\text{Ker } T$.

Given any right inverse \tilde{G} of a surjective linear map $T: \mathcal{F} \rightarrow \mathcal{G}$, the Green’s operator for a regular boundary problem (T, \mathcal{B}) is given by $G = (1 - P)\tilde{G}$, where P is the projector with $\text{Im } P = \text{Ker } T$ and $\text{Ker } P = \mathcal{B}^\perp$. Using this observation, we outline in the following how the Green’s operator can be computed in the ODE setting.

Let (T, \mathcal{B}) be a regular boundary problem for an ODE of order n with $\mathcal{B} = [\beta_1, \dots, \beta_n]$, and let u_1, \dots, u_n be a fundamental system of solutions. We first compute a right inverse of the differential operator T . This can be done by the usual variation-of-constants formula (see for example [13, p. 87] for continuous functions or [14] in a suitable integro-differential algebra setting): Let $W = W(u_1, \dots, u_n)$ be the Wronskian matrix and $d = \det W$. Moreover, let $d_i = \det W_i$, where W_i is the matrix obtained from W by replacing the i th column by the n th unit vector. Then the solution of the initial value problem $Tu = f$, $u(a) = u'(a) = \dots = u^{(n-1)}(a) = 0$ is given by

$$u(x) = \sum_{i=1}^n u_i(x) \int_a^x d_i(\xi) / d(\xi) f(\xi) d\xi. \tag{3}$$

The integral operator $T^\blacklozenge: f \mapsto u$ defined by (3) is a right inverse of T , which we also call the *fundamental right inverse*. Computing the projector $P: \mathcal{F} \rightarrow \mathcal{F}$ with $\text{Im } P = [u_1, \dots, u_n]$ and $\text{Ker } P = [\beta_1, \dots, \beta_n]^\perp$ is a linear algebra problem, see

for example [5, App. A.1]: Let B be the evaluation matrix $B = (\beta_i(u_j))$. Since (T, \mathcal{B}) is regular, B is invertible. Set $(\tilde{\beta}_1, \dots, \tilde{\beta}_n)^t = B^{-1}(\beta_1, \dots, \beta_n)^t$. Then the projector P is given by $u \mapsto \sum_{i=1}^n \tilde{\beta}_i(u) u_i$. Finally, we compute

$$G = (1 - P)T^\blacklozenge \tag{4}$$

to obtain the Green’s operator for (T, \mathcal{B}) .

3 Composing and Factoring Boundary Problems

In this section we discuss the composition of boundary problems corresponding to their Green’s operators. We also describe how factorizations of a boundary problem along a given factorization of the defining operator can be characterized and constructed. We refer again to [5] for further details. In the following, we assume that all operators are defined on suitable spaces such that the composition is well-defined.

Definition 1. We define the composition of boundary problems (T_1, \mathcal{B}_1) and (T_2, \mathcal{B}_2) by $(T_1, \mathcal{B}_1) \circ (T_2, \mathcal{B}_2) = (T_1 T_2, \mathcal{B}_1 \cdot T_2 + \mathcal{B}_2)$.

So the boundary conditions from the first boundary problem are “translated” by the operator from the second problem. The composition of boundary problems is associative but in general not commutative. The next proposition tells us that the composition of boundary problems preserves regularity.

Proposition 1. Let (T_1, \mathcal{B}_1) and (T_2, \mathcal{B}_2) be regular boundary problems with Green’s operators G_1 and G_2 . Then $(T_1, \mathcal{B}_1) \circ (T_2, \mathcal{B}_2) = (T, \mathcal{B})$ is regular with Green’s operator $G_2 G_1$ so that $((T_1, \mathcal{B}_1) \circ (T_2, \mathcal{B}_2))^{-1} = (T_2, \mathcal{B}_2)^{-1} \circ (T_1, \mathcal{B}_1)^{-1}$.

The simplest example of composing two boundary (more specifically, initial value) problems for ODEs is the following. Using the notation from the Introduction, one sees that $(D, [L]) \circ (D, [L]) = (D^2, [LD] + [L]) = (D^2, [L, LD])$.

Next we write the wave equation (2) as $\mathcal{P} = (D_t^2 - D_x^2, [u(x, 0), u_t(x, 0)])$, where $u(x, 0)$ and $u_t(x, 0)$ are short for the functionals $u \mapsto u(x, 0)$ and $u \mapsto u_t(x, 0)$, respectively, with x ranging over \mathbb{R} , and $[\dots]$ denotes the orthogonal closure of the subspace generated by these functionals. For boundary problems with PDEs, we usually have to describe the boundary conditions as the orthogonal closure of some subspaces that we can describe in finite terms. As detailed in [5], we can still compute the composition of two such problems since taking the orthogonal closure commutes with the operations needed for computing the boundary conditions for the composite problem (precomposition with a linear operator and sum of subspaces).

Using this observation, we can compute \mathcal{P} as the composition of the two boundary problems $\mathcal{P}_1 = (D_t - D_x, [u(x, 0)])$ and $\mathcal{P}_2 = (D_t + D_x, [u(x, 0)])$ as follows. By Definition 1, we see that $\mathcal{P}_1 \circ \mathcal{P}_2$ equals

$$(D_t^2 - D_x^2, [u_t(x, 0) + u_x(x, 0)] + [u(x, 0)]) = (D_t^2 - D_x^2, [u(x, 0), u_t(x, 0)]), \tag{5}$$

where the last equality holds since $u(x, 0) = 0$ for $x \in \mathbb{R}$ implies also $u_x(x, 0) = 0$ for $x \in \mathbb{R}$, showing that $u_x(x, 0)$ is in the orthogonal closure $[u(x, 0)]$.

In the following, we assume that for a boundary problem (T, \mathcal{B}) we have a factorization $T = T_1 T_2$ of the defining operator with surjective linear maps T_1, T_2 . In [5], we characterize and construct all factorizations $(T, \mathcal{B}) = (T_1, \mathcal{B}_1) \circ (T_2, \mathcal{B}_2)$ into boundary problems along the given factorization of T . We show in particular that if we factor a regular problem into regular problems, the left factor (T_1, \mathcal{B}_1) is unique, and we can choose for the right factor (T_2, \mathcal{B}_2) any subspace $\mathcal{B}_2 \leq \mathcal{B}$ that makes the problem regular. Moreover, if G_2 is the Green’s operator for some regular right factor (T_2, \mathcal{B}_2) , the boundary conditions for the left factor can be computed by $\mathcal{B}_1 = \mathcal{B} \cdot G_2$. Factoring boundary problems for differential equations allows us to split a problem of higher order into subproblems of lower order, provided we can factor the differential operator. For the latter, we can exploit algorithms and results about factoring ordinary [15,16,17] and partial differential operators [18,19].

For ODEs we can factor boundary problems algorithmically as described in [5] and in an integro-differential algebra setting in [4]. There we assume that we are given a fundamental system of the differential operator T and a right inverse of T_2 . As we will detail in the next paragraph, we can also compute boundary conditions $\mathcal{B}_2 \leq \mathcal{B}$ such that (T_2, \mathcal{B}_2) is a regular right factor, given only a fundamental system of T_2 . We can then compute the left factor as explained above. This can be useful in applications, because it still allows us to factor a boundary problem if we can factor the differential operator and compute a fundamental system of only one factor. The remaining lower order problem can then be solved by numerical methods (and we expect that the integral conditions $\mathcal{B}_1 = \mathcal{B} \cdot G_2$ may be beneficial since they are stable).

Let now (T, \mathcal{B}) be a boundary problem of order $m + n$ with boundary conditions $[\beta_1, \dots, \beta_{m+n}]$. Let $T = T_1 T_2$ be a factorization into factors of respective orders n and m , and let u_1, \dots, u_m be a fundamental system for T_2 . We compute the “partial” $(m + n) \times m$ evaluation matrix $\tilde{B} = \beta_i(u_j)$. Since (T, \mathcal{B}) is regular, the full evaluation matrix is regular and hence the columns of \tilde{B} are linearly independent. Therefore computing the reduced row echelon form yields a regular matrix C such that $C\tilde{B} = \begin{pmatrix} I_m \\ 0 \end{pmatrix}$, where I_m is the $m \times m$ identity matrix. Let now $(\tilde{\beta}_1, \dots, \tilde{\beta}_{m+n})^t = C(\beta_1, \dots, \beta_{m+n})^t$ and $\mathcal{B}_2 = [\tilde{\beta}_1, \dots, \tilde{\beta}_m]$. Then (T_2, \mathcal{B}_2) is a regular right factor since its evaluation matrix is I_m by our construction. See Appendix A for an example.

As a first example, we factor the two-point boundary problem $(D^2, [L, R])$ from the Introduction into two regular problems along the trivial factorization with $T_1 = T_2 = D$. The indefinite integral $A = \int_0^x$ is the Green’s operator for the regular right factor $(D, [L])$. The boundary conditions for the unique left factor are $[LA, RA] = [0, RA] = [RA]$, where $RA = \int_0^1$ is the definite integral. So we obtain $(D, [RA]) \circ (D, [L]) = (D^2, [L, R])$ or in traditional notation

$$\boxed{\begin{matrix} u' = f \\ \int_0^1 u(\xi) d\xi = 0 \end{matrix}} \circ \boxed{\begin{matrix} u' = f \\ u(0) = 0 \end{matrix}} = \boxed{\begin{matrix} u'' = f \\ u(0) = u(1) = 0 \end{matrix}}$$

Note that the boundary condition for the left factor is an integral (Stieltjes) boundary condition.

As an example of a boundary problem for a PDE, we factor the wave equation (2) along the factorization $D_t^2 - D_x^2 = (D_t - D_x)(D_t + D_x)$. In Appendix A, we show that one can use this factorization to determine algorithmically its Green's operator. The boundary problem $\mathcal{P}_2 = (D_t + D_x, [u(x, 0)])$ is a regular right factor. In general, choosing boundary conditions in such a way that they make up a regular boundary problem for a given first-order right factor of a linear PDE amounts to a geometric problem involving the characteristics; compare also Section 4. The Green's operator for \mathcal{P}_2 is $G_2 f(x, t) = \int_{x-t}^x f(\xi, \xi - x + t) d\xi$. We can compute the boundary conditions for the left factor by $[u(x, 0) \cdot G_2, u_t(x, 0) \cdot G_2] = [0, u(x, 0)] = [u(x, 0)]$ so that $\mathcal{P}_1 = (D_t - D_x, [u(x, 0)])$ is the desired left factor. In (5) we have already verified that $\mathcal{P}_1 \circ \mathcal{P}_2 = \mathcal{P}$.

4 Representation of Integro-differential Operators

For representing ordinary boundary problems as well as their Green's operators in a single algebraic structure, we have introduced the algebra of *integro-differential operators* $\mathcal{F}[\partial, \int]$ in [4], see also [14] for a summary. It is based on integro-differential algebras, which bring together the usual derivation structure with a suitable notion of indefinite integration and evaluation. The integro-differential operators are defined as a quotient of the free algebra in the corresponding operators (derivation, integration, evaluation, and multiplication) modulo an infinite parametrized Gröbner basis. See Section 5 for more details and an implementation. Alternatively, integro-differential operators can also be defined directly in terms of normal forms [20].

Let us now turn to the treatment of *partial differential equations*. We are currently forging an adequate notion of integro-differential operators for describing the Green's operators of an interesting class of PDEs, just as $\mathcal{F}[\partial, \int]$ can be used for ODEs. In the remainder of this section we can only give a flavor (and a small test implementation) of how integro-differential operators for PDEs might look like in a simple case that includes the unbounded wave equation (2).

We construct a ring \mathcal{R} of integro-differential operators acting on the function space $\mathcal{F} = C^\infty(\mathbb{R} \times \mathbb{R})$; for simplicity we neglect here the restriction to $\mathbb{R} \times \mathbb{R}_{\geq 0}$. The ring \mathcal{R} is defined as the free \mathbb{C} -algebra in the following indeterminates given with their respective action on a function $f(x, t) \in \mathcal{F}$.

Name	Indeterminates	Action
Differential operators	D_x, D_t	$f_x(x, t), f_t(x, t)$
Integral operators	A_x, A_t	$\int_0^x f(\xi, t) d\xi, \int_0^t f(x, \tau) d\tau$
Evaluation operators	L_x, L_t	$f(0, t), f(x, 0)$
Substitution operators	$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \text{GL}(\mathbb{R}, 2)$	$f(ax + bt, cx + dt)$

Similar to the identities governing $\mathcal{F}[\partial, \int]$, described in [4], various relations among the above operators can now be encoded in a quotient of \mathcal{R} . We will only sketch the most important relations, focusing on those that are needed for the sample computations. (In a more complete setup, the indeterminates should also be chosen in a more economical way. For example, it is possible to subsume the evaluations under the substitutions if one allows all affine transformations by adding translations and singular matrices.)

First of all, we can transfer all relations from $\mathcal{F}[\partial, \int]$ that involve D, A and L , once for the corresponding x -operators and once for the corresponding t -operators. Furthermore, each x -operator commutes with each t -operator. For example, we have $D_x A_x = 1$ but $D_x A_t = A_t D_x$. For normalizing such commutative products, we write the x -operators left of the t -operators. Our strategy for normal forms is thus similar to the case of $\mathcal{F}[\partial, \int]$, the only new ingredient being the substitutions: We will move them to the left as much as possible.

Since substitutions operate on the arguments, it is clear that we must reverse their order when multiplying them as elements of \mathcal{R} . But the most important relations are those that connect the substitutions with the integro-differential indeterminates: The chain rule governs the interaction with differentiation, the substitution rule with integration. While the former gives rise to the identities

$$D_x M = a M D_x + c M D_t \quad \text{and} \quad D_t M = b M D_x + d M D_t$$

for a matrix $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, the relation between M and integrals is a bit subtler. If M is an upper triangular matrix (so that $c = 0$ and $a \neq 0$), the substitution rule yields

$$A_x M = \frac{1}{a}(1 - L_x) M A_x,$$

and if M is a lower triangular matrix (so that $b = 0$ and $d \neq 0$) similarly $A_t M = \frac{1}{d}(1 - L_t) M A_t$.

But there are no such identities for pushing $\begin{pmatrix} 1 & 0 \\ c & 1 \end{pmatrix}$ left of A_x or $\begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix}$ left of A_t ; we leave them in their place for the normal forms. For treating the general case, we make use of a variant of the Bruhat decomposition [21, p. 349], writing $M \in \text{GL}(\mathbb{R}, 2)$ as $\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ c/a & 1 \end{pmatrix} \begin{pmatrix} a & b \\ 0 & (ad-bc)/a \end{pmatrix}$ if $a \neq 0$ and $\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} b & 0 \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ if $a = 0$. Alternatively, we may also use $\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & b/d \\ 0 & 1 \end{pmatrix} \begin{pmatrix} (ad-bc)/d & 0 \\ c & d \end{pmatrix}$ if $d \neq 0$ and $\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} b & a \\ 0 & c \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ if $d = 0$. The former decomposition is applied in deriving the rule for A_x , which reads

$$A_x \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \frac{1}{a} (1 - L_x) \begin{pmatrix} a & b \\ 0 & (ad-bc)/a \end{pmatrix} A_x \begin{pmatrix} 1 & 0 \\ c/a & 1 \end{pmatrix}$$

if $a \neq 0$ and otherwise $A_x \begin{pmatrix} 0 & b \\ c & d \end{pmatrix} = \frac{1}{c} (1 - L_x) \begin{pmatrix} 0 & b \\ c & d \end{pmatrix} A_t$. Analogously, the latter decomposition yields the rule for A_t as

$$A_t \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \frac{1}{d} (1 - L_t) \begin{pmatrix} (ad-bc)/d & 0 \\ c & d \end{pmatrix} A_t \begin{pmatrix} 1 & b/d \\ 0 & 1 \end{pmatrix}$$

if $d \neq 0$ and otherwise $A_t \begin{pmatrix} a & b \\ c & 0 \end{pmatrix} = \frac{1}{b} (1 - L_t) \begin{pmatrix} a & b \\ c & 0 \end{pmatrix} A_x$.

According to the rules above, an \mathcal{R} -operator like $A_x \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix}$ is in normal form. Also $A_x \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix} A_x$ is a normal form, describing an area integral. For interpreting

it geometrically, it is convenient to postmultiply it with the reverse shear, obtaining thus the integral operator $T_k = \begin{pmatrix} 1 & 0 \\ -k & 1 \end{pmatrix} A_x \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix} A_x$. One can easily verify that $T_k f(x, t)$ represents the integral of f taken over the triangle with vertices (x, t) , $(0, y)$ and $(0, t - kx)$. This is the triangle delimited by the y -axis, the horizontal through (x, y) , and the slanted line through (x, t) with slope k . Similar interpretations can be given for products involving A_t .

Finally, we need some rules relating substitutions with evaluations. Here the situation is analogous to the integrals: We can move “most” of the substitutions to the left of an evaluation, but certain shears remain on the right. In detail, we have the rules

$$L_x \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & d \end{pmatrix} L_x \begin{pmatrix} 1 & b/d \\ 0 & 1 \end{pmatrix} \quad \text{if } d \neq 0 \qquad L_x \begin{pmatrix} a & b \\ c & 0 \end{pmatrix} = \begin{pmatrix} 0 & b \\ 1 & 0 \end{pmatrix} L_t \quad \text{otherwise}$$

and

$$L_t \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix} L_t \begin{pmatrix} 1 & 0 \\ c/a & 1 \end{pmatrix} \quad \text{if } a \neq 0 \qquad L_t \begin{pmatrix} 0 & b \\ c & d \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ c & 0 \end{pmatrix} L_x \quad \text{otherwise.}$$

As before, certain products remain as normal forms, for example $L_x \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$. Such an operator acts on a function $f \in \mathcal{F}$ as $f(kt, t)$, collapsing the bivariate function f to the univariate restriction along the diagonal line $x = kt$.

The language of \mathcal{R} -operators is not very expressive, but enough for our modest purposes at this point—expressing the boundary problem (2) and computing its Green’s operator. Let us first look at the general first-order boundary problem with constant coefficients, prescribing homogeneous Dirichlet conditions on an arbitrary line. Fixing the parameters $a, b, c, k \in \mathbb{R}$, it reads as follows:

$$\boxed{\begin{matrix} a u_x + b u_t = f \\ u(kt + c, t) = 0 \end{matrix}} \tag{6}$$

Here $(a, b)^t$ determines the direction (and speed) of the ground characteristics, while $x = kt + c$ gives the line of boundary values. Of course this excludes the horizontal lines $t = \text{const}$, which would have to be treated separately, in a completely analogous manner. Since (in this paper) we are interested only in regular boundary problems, the characteristics must have a transversal intersection with the line of boundary values. Hence we stipulate that $a - kb \neq 0$. Moreover, we will also assume $a \neq 0$; for otherwise one may switch the x - and t -coordinates. A straightforward computation (or a suitable computer algebra system) gives now

$$u(x, y) = \frac{1}{a} \int_X^x f(\xi, \frac{b}{a}(\xi - x) + t) d\xi \quad \text{with} \quad X = \frac{ac + (at - bx)k}{a - bk}.$$

This solution for the general case can be reduced to $(a, b)^t = (1, 0)^t$ and $k = 0$ by first rotating (a, b) into horizontal position, then normalizing it through x -scaling, and finally shearing the line of boundary values into vertical position. This yields the factorization

$$u(x, y) = \begin{pmatrix} 1/K & -k/K \\ -b/L & a/L \end{pmatrix} \cdot \int_{c/K}^x \cdot \begin{pmatrix} a & kL/K \\ b & L/K \end{pmatrix} f(x, y), \tag{7}$$

where $K = a - bk$ and $L = a^2 + b^2$. This is almost an \mathcal{R} -operator, except that we have only allowed $A_x = \int_0^x$ and its t -analog, so we cannot express $\int_{c/K}^x$ unless we allow more evaluations such that we could write the required integral as $A_x - L_x^{c/K} A_x$, where L_x^ξ acts on a function $g(x, y)$ as $g(\xi, y)$.

While it would be straightforward to incorporate such evaluations by adding suitable relations, it is enough for our purposes to restrict the line of boundaries: We require it to pass through the origin so that $c = 0$. In this case we have of course $\int_{c/K}^x = A_x$, and (7) shows that we can indeed write the Green's operator in the \mathcal{R} language.

5 Implementation in Theorema

As explained in Sections 2 and 4, we compute the Green's operator of a boundary problem for an ODE as an *integro-differential operator*. These operators are realized as noncommutative polynomials (introduced by a generic construct for monoid algebras), taken modulo an infinite parametrized Gröbner basis.

As coefficients we allow either standard polynomials or—more generally—exponential polynomials. Informally speaking, an *exponential polynomial* is a linear combination of terms having the form $x^n e^{\lambda x}$, where n is a natural and λ a complex number. Both the standard and the exponential polynomials can again be generated as an instance of the monoid algebra, respectively using \mathbb{N} and $\mathbb{N} \times \mathbb{C}$ as a term monoid. In this way, we have complete algorithmic control over the coefficient functions (modulo Mathematica's simplifier for constants); see also [22]. Alternatively, we can also take as coefficients all functions representable in Mathematica and let it do the operations on them.

We describe now briefly the representation of integro-differential operators and the implementation of the main algorithms solving, composing and factoring boundary problems. The implementation will soon be available at the website www.theorema.org. It is based on Theorema [6], a system designed as an integrated environment for doing mathematics, in particular proving, computing, and solving in various domains of mathematics. Its core language is higher-order predicate logic, containing a natural *programming language* such that algorithms can be coded and verified in a unified formal frame.

We make heavy use of *functors*, introduced and first implemented in Theorema by Buchberger. The general idea—and its use for structuring those domains in which Gröbner bases can be computed—is described in [23,24], where one can also find references to original and early papers by Buchberger on the subject. For a general discussion of functor programming, see also [25].

Functors are a powerful tool for building up *hierarchical domains* in mathematics in a modular and generic way that unites elegance and formal clarity. In Theorema, the notion of a functor is akin to functors in ML, not to be confused with the functors of category theory. From a computational point of view, a Theorema functor is a higher-order function that produces a new domain (carrier and operations) from given domains: operations in the new domain are defined in terms of operations in the underlying domains. Apart from this computational

aspect, functors also have an important reasoning aspect—a functor transports properties of the input domains to properties of the output domain, for example by “conservation theorems”.

The `MonoidAlgebra` is the crucial functor that builds up *polynomials*, starting from the base categories of fields with an ordering and ordered monoids. We construct first the free vector space V over a field K generated by the set of words in an ordered monoid W via the functor `FreeVecSpc`[K, W]. Then we extend this domain by introducing a multiplication using the corresponding operations in K and W as follows.

```

MonoidAlgebra[ $K, W$ ] = where [  $V = \text{FreeVecSpc}[K, W]$  ,
Functor [  $P, \text{any}[c, d, f, g, \xi, \eta, \bar{m}, \bar{n}]$  ,
   $s = \langle \rangle$ 
  ...(* linear operations from V *)
  (* multiplication *)
   $\langle \rangle *_{\mathcal{P}} g = \langle \rangle$ 
   $f *_{\mathcal{P}} \langle \rangle = \langle \rangle$ 
   $\langle \langle c, \xi \rangle, \bar{m} \rangle *_{\mathcal{P}} \langle \langle d, \eta \rangle, \bar{n} \rangle = \left( \left( c *_{\mathcal{K}} d, \xi *_{\mathcal{W}} \eta \right) \right)_{\mathcal{P}} + \langle \langle c, \xi \rangle \rangle *_{\mathcal{P}} \langle \bar{n} \rangle + \langle \bar{m} \rangle *_{\mathcal{P}} \langle \langle d, \eta \rangle, \bar{n} \rangle$ 
]]

```

For building up the *integro-differential operators* over an integro-differential algebra \mathcal{F} of coefficient functions, `FreeIntDiffOp`[\mathcal{F}, K] constructs an instance of the monoid algebra with the word monoid over the infinite alphabet consisting of the letters ∂ and \int along with a basis of \mathcal{F} and all multiplicative characters corresponding to evaluations at points in K .

```

Definition["IntDiffOp", any [ $\mathcal{F}, K$ ],
IntDiffOp [ $\mathcal{F}, K$ ] = where [ $\mathcal{A} = \text{FreeIntDiffOp}[\mathcal{F}, K]$  ,  $\mathcal{G} = \text{GreenSystem}[\mathcal{F}, K]$ 
  QuotAlg [GBNF [ $\mathcal{A}, \mathcal{G}$ ]]]
]

```

The `GreenSystem` functor contains the encoding of the rewrite system described in Table 1 of [4,14], representing a noncommutative Gröbner basis. The normal forms with respect to total reduction modulo infinite Gröbner bases are introduced in the `GBNF` functor, while the `QuotAlg` functor creates the quotient algebra from the corresponding canonical simplifier.

In Appendix A, we present a few examples of boundary problems for ODEs whose *Green’s operators* are computed using (4), which now takes on the following concrete form in Theorema code.

```

GreensOp [ $F, \mathcal{B}$ ] =  $\left( \mathbf{1}_{\mathcal{A}} - \text{Proj}[\mathcal{B}, F] \right) *_{\mathcal{A}} \text{RightInv}[F]_{\mathcal{P}}$ 

```

Here \mathcal{B} is the vector of boundary conditions and F the given fundamental system of solutions.

In a way similar to the integro-differential operators $\mathcal{F}[\partial, \int]$ for ODEs, we have also implemented the integro-differential operators \mathcal{R} for the simple PDE setting outlined in Section 4. Using the same functor hierarchy, we added the corresponding rules for the operators $D_x, D_t, A_x, A_t, L_x, L_t$ and the substitution operators defined by matrices in $GL(\mathbb{R}, 2)$. Moreover, we implemented the computation of Green’s operators for first-order boundary problems (7). With the

factorization (5) we can then compute the Green's operator for the unbounded wave equation (Appendix A).

6 Conclusion

The implementation of our symbolic framework for boundary problems allows us in particular to *solve boundary problems* for *ODEs* from a given fundamental system of the corresponding homogeneous equations. Given a factorization of the differential operator and a fundamental system of one of the factors, we can also *factor boundary problems* into lower order problems. In both cases it would be interesting to investigate the combination with numerical approaches to differential equations and boundary problems. For example, how can we use a fundamental system coming from a numerical algorithm or how can numerical methods be adapted to deal with integral boundary conditions?

The current setting for *PDEs* is of course still very limited and should only be seen as a starting point for future work. But in combination with our factorization approach, we believe that it can be extended to include more complicated problems. For example, the wave equation on the *bounded interval* $[0, 1]$, which in our notation reads as $\mathcal{P} = (D_t^2 - D_x^2, [u(x, 0), u_t(x, 0), u(0, t), u(1, t)])$ with x ranging over $[0, 1]$ and t over $\mathbb{R}_{\geq 0}$, can be factored (5) into $\mathcal{P} = \mathcal{P}_1 \circ \mathcal{P}_2$ with

$$\mathcal{P}_1 = (D_t - D_x, [u(x, 0), \int_{\max(1-t, 0)}^1 u(\xi, \xi + t - 1) d\xi])$$

and $\mathcal{P}_2 = (D_t + D_x, [u(x, 0), u(0, t)])$. The more complicated structure of the Green's operator for \mathcal{P} (it involves a finite sum with an upper bound depending on its argument) is reflected in the Green's operator for the left factor \mathcal{P}_1 . Its computation leads in this case to a simple functional equation, but a systematic approach to compute and represent Green's operators for PDEs with *integral boundary conditions* still needs to be developed. In a generalized setting including the bounded wave equation, we would also have to allow for more complicated geometries: as a first step bounded intervals and then also arbitrary convex sets.

References

1. Stakgold, I.: Green's functions and boundary value problems. John Wiley & Sons, New York (1979)
2. Rosenkranz, M., Buchberger, B., Engl, H.W.: Solving linear boundary value problems via non-commutative Gröbner bases. Appl. Anal. 82, 655–675 (2003)
3. Rosenkranz, M.: A new symbolic method for solving linear two-point boundary value problems on the level of operators. J. Symbolic Comput. 39, 171–199 (2005)
4. Rosenkranz, M., Regensburger, G.: Solving and factoring boundary problems for linear ordinary differential equations in differential algebras. J. Symbolic Comput. 43, 515–544 (2008)
5. Regensburger, G., Rosenkranz, M.: An algebraic foundation for factoring linear boundary problems. Ann. Mat. Pura Appl. 188(4), 123–151 (2009)

6. Buchberger, B., Craciun, A., Jebelean, T., Kovacs, L., Kutsia, T., Nakagawa, K., Piroi, F., Popov, N., Robu, J., Rosenkranz, M., Windsteiger, W.: Theorema: Towards computer-aided mathematical theory exploration. *J. Appl. Log.* 4, 359–652 (2006)
7. Buchberger, B.: An algorithm for finding the bases elements of the residue class ring modulo a zero dimensional polynomial ideal (German). PhD thesis, Univ. of Innsbruck (1965); English translation *J. Symbolic Comput.* 41(3-4), 475–511 (2006)
8. Buchberger, B.: Introduction to Gröbner bases. In: Buchberger, B., Winkler, F. (eds.) *Gröbner bases and applications*, Cambridge Univ. Press, Cambridge (1998)
9. Mora, T.: An introduction to commutative and noncommutative Gröbner bases. *Theoret. Comput. Sci.* 134, 131–173 (1994)
10. Köthe, G.: *Topological vector spaces*, vol. I. Springer, New York (1969)
11. Brown, R.C., Krall, A.M.: Ordinary differential operators under Stieltjes boundary conditions. *Trans. Amer. Math. Soc.* 198, 73–92 (1974)
12. Kamke, E.: *Differentialgleichungen. Lösungsmethoden und Lösungen. Teil I: Gewöhnliche Differentialgleichungen.* Akademische Verlagsgesellschaft, Leipzig (1967)
13. Coddington, E.A., Levinson, N.: *Theory of ordinary differential equations.* McGraw-Hill Book Company, Inc., New York (1955)
14. Rosenkranz, M., Regensburger, G.: Integro-differential polynomials and operators. In: Jeffrey, D. (ed.) *Proceedings of ISSAC 2008*, pp. 261–268. ACM, New York (2008)
15. van der Put, M., Singer, M.F.: *Galois theory of linear differential equations.* Springer, Berlin (2003)
16. Schwarz, F.: A factorization algorithm for linear ordinary differential equations. In: *Proceedings of ISSAC 1989*, pp. 17–25. ACM, New York (1989)
17. Tsarev, S.P.: An algorithm for complete enumeration of all factorizations of a linear ordinary differential operator. In: *Proceedings of ISSAC 1996*, pp. 226–231. ACM, New York (1996)
18. Grigoriev, D., Schwarz, F.: Loewy- and primary decompositions of D-modules. *Adv. in Appl. Math.* 38, 526–541 (2007)
19. Tsarev, S.P.: Factorization of linear partial differential operators and Darboux integrability of nonlinear PDEs. *SIGSAM Bull.* 32, 21–28 (1998)
20. Regensburger, G., Rosenkranz, M., Middeke, J.: A skew polynomial approach to integro-differential operators. In: *Proceedings of ISSAC 2009*. ACM, New York (to appear, 2009)
21. Cohn, P.M.: *Further algebra and applications.* Springer, London (2003)
22. Buchberger, B., Regensburger, G., Rosenkranz, M., Tec, L.: General polynomial reduction with Theorema functors: Applications to integro-differential operators and polynomials. *ACM Commun. Comput. Algebra* 42, 135–137 (2008)
23. Buchberger, B.: Groebner rings and modules. In: Maruster, S., Buchberger, B., Negru, V., Jebelean, T. (eds.) *Proceedings of SYNASC 2001*, pp. 22–25 (2001)
24. Buchberger, B.: Groebner bases in Theorema using functors. In: Faugere, J., Wang, D. (eds.) *Proceedings of SCC 2008*, pp. 1–15. LMIB Beihang University Press (2008)
25. Windsteiger, W.: Building up hierarchical mathematical domains using functors in Theorema. *Electr. Notes Theor. Comput. Sci.* 23, 401–419 (1999)

A Sample Computations

Let us again consider example (II). By our implementation, we obtain the Green's operator for the boundary problem with the corresponding Green's function. As noted in [3], the Green's function provides a canonical form for the Green's operator. In the following, we use the notation $Au = \int_0^x u(\xi) d\xi$, $Bu = \int_x^1 u(\xi) d\xi$, $Lu = u(0)$, $Ru = u(1)$, and $A1f(x, t) = \int_0^x f(\xi, t) d\xi$.

```

Compute[AsGreen[GreensOp[D2, <<{1, <<"|]", 0}>>>, <{1, <<"|]", 1}>>>]]]
-A x - x B + x A x + x B x

Compute[GreensFct[GreensOp[D2, <<{1, <<"|]", 0}>>>, <{1, <<"|]", 1}>>>]]]
{
  -ξ + x ξ ← ξ ≤ x
  -x + x ξ ← x < ξ
}
    
```

As explained in Section 3, we can factor (II) along a factorization of the differential operator, given a fundamental system for the right factor. Here is how we can compute the boundary conditions of the left and right factor problems, respectively.

```

Compute[AsGreen[Factorize[D, D, <<{1, <<"|]", 0}>>>, <{1, <<"|]", 1}>>>, <<{1, <>>>]]]
<<{A + B}, <L>>
    
```

We consider as a second example the fourth order boundary problem [4, Ex. 33]:

$$\begin{cases} u'''' + 4u = f, \\ u(0) = u(1) = u'(0) = u'(1) = 0. \end{cases} \tag{8}$$

Factoring the boundary problem along $D^4 + 4 = (D^2 - 2i)(D^2 + 2i)$, we obtain the following boundary conditions for the factor problems.

```

Compute[AsGreen[Factorize[D2 - 2 i, D2 + 2 i,
  <<{1, <<"|]", 0}>>>, <{1, <<"|]", 1}>>>, <{1, <<"|]", 0}, "θ">>>, <{1, <<"|]", 1}, "θ">>>,
  <<{1, <<"|]", <0, -1 + i}>>>>, <{1, <<"|]", <0, 1 + (-1) i}>>>>]]]
<<{A e(Complex[-1,1] x) + B e(Complex[-1,1] x), A e(Complex[1,-1] x) + B e(Complex[1,-1] x), <L, R}>>
    
```

With our implementation we can also compute its Green's operator and verify the solution presented in [4].

The final example for ODEs is a third order boundary problem with exponential coefficients.

$$\begin{cases} u''' - (e^x + 2)u'' - u' + (e^x + 2)u = f, \\ u(0) = u(1) = u'(1) = 0. \end{cases} \tag{9}$$

Here we use as coefficient algebra all functions representable in Mathematica. The Green's operator is computed as follows.

$$\begin{aligned}
 & \text{Compute} \left[\text{GreensOp} \left[\left(\langle \langle 1, \text{mma}[e^x] \rangle \rangle, \langle \langle 1, \text{mma}[e^{-x}] \rangle \rangle, \langle \langle -1, \text{mma}[e^x e^{-x}] \rangle \rangle, \langle 1, \text{mma}[e^{e^x}] \rangle \right) \right], \right. \\
 & \quad \left. \langle \langle 1, \langle \langle "[]", 0 \rangle \rangle \rangle, \langle \langle 1, \langle \langle "[]", 1 \rangle \rangle \rangle, \langle \langle 1, \langle \langle "[]", 1 \rangle, " \partial " \rangle \rangle \rangle \right] \\
 & (-1 + e)^{-2} e^{-1} e^{e^x} A + (-1 + e)^{-2} e^{-1} e^{e^x} B + (-1) (-1 + e)^{-2} e^{-1} e^{e^{e^x + (-1) \times} A} + \\
 & (-1) (-1 + e)^{-2} e^{-1} e^{e^{e^x + (-1) \times} B} + \left(\frac{1}{2} + \frac{1}{2} (-1 + e)^{-2} \right) e^{-1 \times} A + \frac{1}{2} (-1 + e)^{-2} e^{-1 \times} B + \\
 & \left(\frac{-1}{2} \right) (-1 + e)^{-2} e^x A + \left(\frac{-1}{2} \right) (-1 + e)^{-2} e^x B + (-1 + e)^{-2} e^{-1} e^{e^x} A e^{-2 \times} + \\
 & (-2) (-1 + e)^{-2} e^{-1} e^{e^x} A e^{-1 \times} + (-1) e^{e^x} B e^{-1 e^x + (-2) \times} + (-1 + e)^{-2} e^{-1} e^{e^x} B e^{-2 \times} + \\
 & (-2) (-1 + e)^{-2} e^{-1} e^{e^x} B e^{-1 \times} + (-1) (-1 + e)^{-2} e^{-1} e^{e^{e^x + (-1) \times} A} e^{-2 \times} + 2 (-1 + e)^{-2} e^{-1} e^{e^{e^x + (-1) \times} A} e^{-1 \times} + \\
 & e^{e^{e^x + (-1) \times} B} e^{-1 e^x + (-2) \times} + (-1) (-1 + e)^{-2} e^{-1} e^{e^{e^x + (-1) \times} B} e^{-2 \times} + 2 (-1 + e)^{-2} e^{-1} e^{e^{e^x + (-1) \times} B} e^{-1 \times} + \\
 & \left(1 + \frac{1}{2} (-1 + (-1 + e)^{-2}) \right) e^{-1 \times} A e^{-2 \times} + (-1 + (-1) (-1 + e)^{-2}) e^{-1 \times} A e^{-1 \times} + \\
 & \frac{1}{2} (-1 + (-1 + e)^{-2}) e^{-1 \times} B e^{-2 \times} + (-1) (-1 + e)^{-2} e^{-1 \times} B e^{-1 \times} + \left(\frac{-1}{2} + \frac{1}{2} (-2 + e) (-1 + e)^{-2} e \right) e^x A e^{-2 \times} + \\
 & (-1 + e)^{-2} e^x A e^{-1 \times} + \frac{1}{2} (-2 + e) (-1 + e)^{-2} e e^x B e^{-2 \times} + (-1 + e)^{-2} e^x B e^{-1 \times}
 \end{aligned}$$

As a last example, we return to the boundary problem for the *wave equation* (2). With Proposition 1 and using the factorization (5), we can compute the Green’s operator for (2) simply by composing the Green’s operators of the first-order problems $\mathcal{P}_1 = (D_t - D_x, [u(x, 0)])$ and $\mathcal{P}_2 = (D_t + D_x, [u(x, 0)])$. Relative to the setting in Section 4, we switch the x - and t -coordinates.

$$\begin{aligned}
 & \text{Compute} \left[\text{GreensOp} [1, -1, 0] \star \text{GreensOp} [1, 1, 0] \right] \\
 & \langle \langle 1, \langle \langle \text{mat}, \langle \langle 1, 0 \rangle, \langle -1, 1 \rangle \rangle \rangle, A1, \langle \langle \text{mat}, \langle \langle 1, 0 \rangle, \langle 2, 1 \rangle \rangle \rangle, A1, \langle \langle \text{mat}, \langle \langle 1, 0 \rangle, \langle -1, 1 \rangle \rangle \rangle \rangle \rangle
 \end{aligned}$$

Interchanging again t and x , this corresponds in the usual notation to $G_1 f(x, t) = \int_0^t f(\xi, -\xi + x + t) d\xi$ and $G_2 f(x, t) = \int_0^t f(\xi, \xi + x - t) d\xi$, which yields

$$G_2 G_1 f(x, t) = \int_0^t \int_0^\tau f(\xi, 2\tau - \xi + x - t) d\xi d\tau$$

for the Green’s operator of the unbounded wave equation (2).

Mathematical Model for Dengue Epidemics with Differential Susceptibility and Asymptomatic Patients Using Computer Algebra

Clarita Saldarriaga Vargas

Logic and Computation Group
Program of Physical Engineering
School of Sciences and Humanities
EAFIT University
Medellín, Colombia
csalda13@eafit.edu.co

Abstract. When there are diseases affecting large populations where the social, economic and cultural diversity is significant within the same region, the biological parameters that determine the behavior of the dispersion disease analysis are affected by the selection of different individuals. Therefore and because of the variety and magnitude of the communities at risk of contracting dengue disease around all over the world, suggest defining differentiated populations with individual contributions in the results of the dispersion dengue disease analysis. In this paper those conditions were taken in account when several epidemiologic models were analyzed. Initially a stability analysis was done for a SEIR mathematical model of Dengue disease without differential susceptibility. Both free disease and endemic equilibrium states were found in terms of the basic reproduction number and were defined in the Theorem (3.1). Then a DSEIR model was solved when a new susceptible group was introduced to consider the effects of important biological parameters of non-homogeneous populations in the spreading analysis. The results were compiled in the Theorem (3.2). Finally Theorems (3.3) and (3.4) resumed the basic reproduction numbers for three and n different susceptible groups respectively, giving an idea of how differential susceptibility affects the equilibrium states. The computations were done using an algorithmic method implemented in Maple 11, a general-purpose computer algebra system.

Keywords: Dengue disease, epidemiology, SEIR model, DSEIR model, differential susceptibility, basic reproduction number, equilibrium states, stability analysis.

1 Introduction

Dengue disease is one of the most common mosquito-transmitted viral diseases, caused by four virus serotypes of the genus *Flavivirus* with high occurrence in tropical and subtropical regions, including urban districts and rural areas. Some

of the disease signs and symptoms are high fever, severe headache, muscle and joint pains and red rash. Currently there are about 2.5 billion people at risk from dengue disease in over 100 endemic countries and is estimated that there may be 50 million cases of dengue infection worldwide every year (Figure 1). In consonance with that, mathematical models of infectious diseases have been done to approximate the future course of an outbreak and evaluate strategies to control an epidemic.

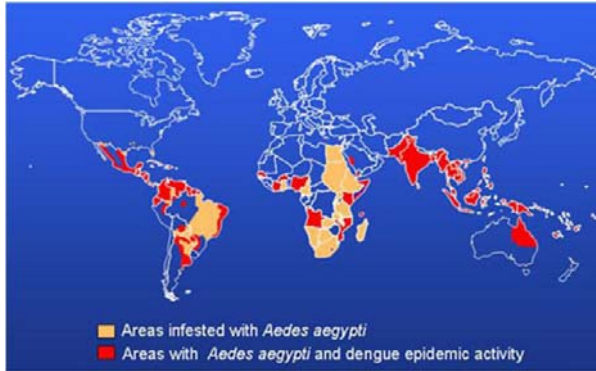


Fig. 1. World distribution of Dengue viruses and their mosquito vector, *Aedes aegypti*, in 2005

When there are diseases that affect large populations, where the social, economic and cultural diversity is significant within the same region, the biological parameters that determine the behavior of the dispersion disease analysis are affected by the selection of different individuals. As mentioned earlier, Dengue is a disease that affects all sorts of people located in tropical and subtropical regions. Communities on every continent are in constant risk of contracting dengue through mosquito bites and therefore of developing the disease. The variety and magnitude of the populations surrounding the earth along the parallel of the equator, suggest defining differentiated populations with particular contributions in the results of the dispersion disease analysis. To achieve this, below are several epidemiological models that have synthesized their results in a series of theorems. At first we started to establish a homogeneous distribution of communities in which there is only one type of susceptible people. Then we define two and three to n different types of susceptible populations which are characterized by having three distinct parameters for each. Those sorts of models are described below.

Compartmental SEIR models divide the human population in four classes: susceptible, symptomatic infectious, asymptomatic infectious and recovered, being a good spreading dengue disease approaches when large human populations with homogeneous behavior are supposed as one susceptible compartment.

Nevertheless, when a non-homogeneous distribution of population is a potential susceptible to acquire the disease, the SEIR model without differential susceptibility isn't able to take in account specific features and biological parameters with high importance for the mathematical spreading analysis. Heterogeneous mixing of the population because of economic, cultural and social tendencies, disease location, rural-urban migration and urban infrastructure are some of the factors that impact in the parameter values of spreading analysis.

For this case is important to use a DSEIR model that includes more than one susceptible group, in which new different susceptible classes were introduced. In this paper a multiple differential susceptibility were given by a DSEIR model and three particular parameters were used to distinguish each susceptible group. At the beginning a stability analysis were done for a SEIR mathematical model without differential susceptibility. Both free disease and endemic equilibrium states were found in terms of the basic reproduction number and were defined in the Theorem (3.1). Then a DSEIR model was solved when a new susceptible group was introduced to consider the effects of important biological parameters of non-homogeneous populations in the spreading analysis. The results were compiled in the Theorem (3.2). Finally Theorems (3.3) and (3.4) resumed the basic reproduction numbers for three and n different susceptible groups respectively. The computations were done using an algorithmic method [23] implemented in Maple 11, based on biological and mathematical criteria for determining the equilibrium states.

2 Problem

Mathematical model without differential susceptibility

To study the mathematical SEIR model of spread of dengue disease, we need to define four classes of human populations and two classes of vector population. Those are the susceptible, asymptomatic infected, symptomatic infected and recovered human population and for vector the susceptible and infected population. In this first model we will not define different kinds of susceptible population, like we will do in the second one. Vector population must be divided only in two groups because it never recovers after the infection. Each group or compartment is named below:

$X(t)$ represents the number of individuals who are not yet infected with the disease at time t , or those who are susceptible to the disease,

$Y(t)$ is the number of asymptomatic infectious person (people who have been infected with the disease but don't show signs of symptoms) at time t ,

$Z(t)$ is the number of symptomatic infectious person (people who have been infected with the disease and show signs of symptoms) at time t ,

$W(t)$ is the number of recovered person at time t . Those in this category have been infected but then recovered from the disease. They are not able to be infected again or to transmit the infection to others.

$X_v(t)$ is the number of susceptible vector populations at time t ,

$Y_v(t)$ is the number of infectious vector population at time t .

Both symptomatic and asymptomatic infectious humans are capable of spreading the disease to those in the susceptible category. The dynamic model for each compartment of human and vector populations without differential susceptibility for dengue disease [2] is given by:

$$\frac{d}{dt}X(t) = \rho N_h - \frac{b(\beta_{ha} + \beta_{hs})X(t)Z_v(t)}{N_h} - \mu_h X(t) \tag{1}$$

$$\frac{d}{dt}Y(t) = \frac{b\beta_{ha}X(t)Z_v(t)}{N_h} - (\mu_h + r)Y(t) \tag{2}$$

$$\frac{d}{dt}Z(t) = \frac{b\beta_{hs}X(t)Z_v(t)}{N_h} - (\mu_h + r)Z(t) \tag{3}$$

$$\frac{d}{dt}W(t) = r(Y(t) + Z(t)) - \mu_h W(t) \tag{4}$$

$$\frac{d}{dt}X_v(t) = C - \frac{b\beta_v X_v(t)(Y(t) + Z(t))}{N_h} - \mu_v X_v(t) \tag{5}$$

$$\frac{d}{dt}Z_v(t) = \frac{b\beta_v X_v(t)(Y(t) + Z(t))}{N_h} - \mu_v Z_v(t) \tag{6}$$

where

- N_t is the total number of human population,
- N_v is the total number of vector population,
- β_{ha} is the transmission probability of dengue virus from vector population to human population and become asymptomatic infectious human population,
- β_{hs} is the transmission probability of dengue virus from vector population to human population and become symptomatic infectious human population,
- μ_h is the death rate of the human population,
- μ_v is the death rate of the vector population,
- ρ is the birth rate of the human population,
- b is the biting rate of the vector population,
- r is the recovery rate of the human population,
- C is the constant recruitment rate of the vector population.

Now, knowing that $X(t)$, $Y(t)$, $Z(t)$ and $W(t)$ are fractions of the total human population, we can say that

$$x(t) + y(t) + z(t) + w(t) = 1$$

where

$$x(t) = \frac{X(t)}{N_h}, y(t) = \frac{Y(t)}{N_h}, z(t) = \frac{Z(t)}{N_h}, w(t) = \frac{W(t)}{N_h}$$

are the compartment densities.

The same is done for the vector compartments:

$$x_v(t) + y_v(t) = 1$$

where

$$x_v(t) = \frac{X_v(t)}{N_v}, y_v(t) = \frac{Y_v(t)}{N_v}$$

Using the last four equations we can resume the system (1-6) in the next expressions:

$$\begin{aligned} \left(\frac{d}{dt}x(t)\right) N_h &= \rho N_h - bx(t)z_v(t)N_v\beta_{ha} \\ &\quad -bx(t)z_v(t)N_v\beta_{hs} - \mu_h x(t)N_h \end{aligned} \tag{7}$$

$$\left(\frac{d}{dt}y(t)\right) N_h = bx(t)z_v(t)N_v\beta_{ha} - y(t)N_h\mu_h - y(t)N_hr \tag{8}$$

$$\left(\frac{d}{dt}z(t)\right) N_h = bx(t)z_v(t)N_v\beta_{hs} - z(t)N_h\mu_h - z(t)N_hr \tag{9}$$

$$\begin{aligned} \left(\frac{d}{dt}z_v(t)\right) N_v &= N_v b\beta_v y(t) - N_v b\beta_v y(t)z_v(t) + N_v b\beta_v z(t) \\ &\quad - N_v b\beta_v z(t)z_v(t) - \mu_v z_v(t)N_v \end{aligned} \tag{10}$$

Mathematical model with differential susceptibility

In order to make and analyze the mathematical model for susceptible populations who have different decisive biological parameters for the spreading disease results, we should introduce another group of this sort of population to take in account its effects and optimize the mathematical model of dengue disease.

For that, we denote as $X_1(t)$ and $X_2(t)$ the new susceptible groups with $\beta_{ha,1}, \beta_{hs,1}$ and ρ_1 for $X_1(t)$ and $\beta_{ha,2}, \beta_{hs,2}$, and ρ_2 for $X_2(t)$. The rest parameters are the same for both susceptible groups.

The dynamic model for each compartment including the differential susceptibility groups [4,5,6] becomes as it is shown:

$$\frac{d}{dt}X_1(t) = \rho_1 N_h - \frac{b(\beta_{ha,1} + \beta_{hs,1}) X_1(t) Z_v(t)}{N_h} - \mu_h X_1(t) \tag{11}$$

$$\frac{d}{dt}X_2(t) = \rho_2 N_h - \frac{b(\beta_{ha,2} + \beta_{hs,2}) X_2(t) Z_v(t)}{N_h} - \mu_h X_2(t) \tag{12}$$

$$\frac{d}{dt}Y(t) = \frac{b\beta_{ha,1} X_1(t) Z_v(t)}{N_h} + \frac{b\beta_{ha,2} X_2(t) Z_v(t)}{N_h} - (\mu_h + r) Y(t) \tag{13}$$

$$\frac{d}{dt}Z(t) = \frac{b\beta_{hs,1} X_1(t) Z_v(t)}{N_h} + \frac{b\beta_{hs,2} X_2(t) Z_v(t)}{N_h} - (\mu_h + r) Z(t) \tag{14}$$

$$\frac{d}{dt}W(t) = r(Y(t) + Z(t)) - \mu_h W(t) \tag{15}$$

$$\frac{d}{dt}X_v(t) = C - \frac{b\beta_v X_v(t) (Y(t) + Z(t))}{N_h} - \mu_v X_v(t) \tag{16}$$

$$\frac{d}{dt}Z_v(t) = \frac{b\beta_v X_v(t) (Y(t) + Z(t))}{N_h} - \mu_v Z_v(t) \tag{17}$$

The compartments densities satisfies that

$$x_1(t) + x_2(t) + y(t) + z(t) + w(t) = 1$$

where

$$x_1(t) = \frac{X_1(t)}{N_h}, x_2(t) = \frac{X_2(t)}{N_h}, y(t) = \frac{Y(t)}{N_h}, z(t) = \frac{Z(t)}{N_h}, w(t) = \frac{W(t)}{N_h}$$

are the compartment densities for the model with differential susceptibility.

The same follows for the vector compartments:

$$x_v(t) + y_v(t) = 1$$

where

$$x_v(t) = \frac{X_v(t)}{N_v}, y_v(t) = \frac{Y_v(t)}{N_v}$$

Using the last four equations we can resume the system (11-17) in the next expressions:

$$\begin{aligned} \left(\frac{d}{dt}x_1(t)\right)N_h &= \rho_1N_h - bx_1(t)z_v(t)N_v\beta_{ha,1} \\ &\quad -bx_1(t)z_v(t)N_v\beta_{hs,1} - \mu_hx_1(t)N_h \end{aligned} \quad (18)$$

$$\begin{aligned} \left(\frac{d}{dt}x_2(t)\right)N_h &= \rho_2N_h - bx_2(t)z_v(t)N_v\beta_{ha,2} \\ &\quad -bx_2(t)z_v(t)N_v\beta_{hs,2} - \mu_hx_2(t)N_h \end{aligned} \quad (19)$$

$$\begin{aligned} \left(\frac{d}{dt}y(t)\right)N_h &= bx_1(t)z_v(t)N_v\beta_{ha,1} + bx_2(t)z_v(t)N_v\beta_{ha,2} \\ &\quad -y(t)N_h\mu_h - y(t)N_hr \end{aligned} \quad (20)$$

$$\begin{aligned} \left(\frac{d}{dt}z(t)\right)N_h &= bx_1(t)z_v(t)N_v\beta_{hs,1} + bx_2(t)z_v(t)N_v\beta_{hs,2} \\ &\quad -z(t)N_h\mu_h - z(t)N_hr \end{aligned} \quad (21)$$

$$\begin{aligned} \left(\frac{d}{dt}z_v(t)\right)N_v &= N_vb\beta_vy(t) - N_vb\beta_vy(t)z_v(t) + N_vb\beta_vz(t) \\ &\quad -N_vb\beta_vz(t)z_v(t) - \mu_vz_v(t)N_v \end{aligned} \quad (22)$$

3 Results

3.1 Mathematical Model without Differential Susceptibility

Theorem 3.1. *For the model without differential susceptibility given by the system (7-10) the following properties hold when host and vector populations with constant size are assumed:*

1. *The equilibrium point without infection, called free disease equilibrium is given by*

$$x = \frac{\rho}{\mu_h} = 1, y = 0, z = 0, w = 0$$

2. The equilibrium point with infection, called endemic equilibrium is given by

$$\begin{aligned}
 x &= \frac{N_h (b\beta_v \rho + \mu_h \mu_v + \mu_v r)}{b\beta_v (\mu_h N_h + bN_v \beta_{hs} + bN_v \beta_{ha})} \\
 y &= \frac{(b^2 \beta_v \rho N_v \beta_{hs} + b^2 \beta_v \rho N_v \beta_{ha} - \mu_h^2 N_h \mu_v - \mu_h N_h \mu_v r) \beta_{ha}}{b\beta_v (\mu_h N_h + bN_v \beta_{hs} + bN_v \beta_{ha}) (\beta_{ha} + \beta_{hs}) (\mu_h + r)} \\
 z &= \frac{(b^2 \beta_v \rho N_v \beta_{hs} + b^2 \beta_v \rho N_v \beta_{ha} - \mu_h^2 N_h \mu_v - \mu_h N_h \mu_v r) \beta_{hs}}{b\beta_v (\mu_h N_h + bN_v \beta_{hs} + bN_v \beta_{ha}) (\beta_{ha} + \beta_{hs}) (\mu_h + r)} \\
 w &= \frac{b^2 \beta_v \rho N_v \beta_{hs} + b^2 \beta_v \rho N_v \beta_{ha} - \mu_h^2 N_h \mu_v - \mu_h N_h \mu_v r}{(\beta_{ha} + \beta_{hs}) N_v (b\beta_v \rho + \mu_h \mu_v + \mu_v r) b}
 \end{aligned}$$

3. The free disease equilibrium point is locally stable when

$$R_0 < 1$$

where

$$R_0 = \frac{\beta_v C b^2 (\beta_{ha} + \beta_{hs})}{\mu_v^2 N_h (\mu_h + r)}$$

is the basic reproductive number for dengue in the model without differential susceptibility.

4. The endemic equilibrium is locally stable when exists, it is when

$$1 < R_0$$

Proof

1. Is directly obtained by computation.
2. Is directly obtained by computation.
3. The jacobian for the system (7-10) is given by

$$\begin{bmatrix}
 A & 0 & 0 & B \\
 bwN_v\beta_{ha} & -\mu_h N_h - N_h r & 0 & bxN_v\beta_{ha} \\
 bwN_v\beta_{hs} & 0 & -\mu_h N_h - N_h r & bxN_v\beta_{hs} \\
 0 & N_v b\beta_v - N_v b\beta_v w & N_v b\beta_v - N_v b\beta_v w & C
 \end{bmatrix}$$

where A, B and C are given by

$$A = -bwN_v\beta_{ha} - bwN_v\beta_{hs} - \mu_h N_h$$

$$B = -bxN_v\beta_{ha} - bxN_v\beta_{hs}$$

$$C = -N_v b\beta_v y - N_v b\beta_v z - \mu_v N_v$$

and when this jacobian is evaluated at the free-disease equilibrium point we obtain

$$\begin{bmatrix} -\mu_h N_h & 0 & 0 & -\frac{b\rho N_v\beta_{ha}}{\mu_h} - \frac{b\rho N_v\beta_{hs}}{\mu_h} \\ 0 & -\mu_h N_h - N_h r & 0 & \frac{b\rho N_v\beta_{ha}}{\mu_h} \\ 0 & 0 & -\mu_h N_h - N_h r & \frac{b\rho N_v\beta_{hs}}{\mu_h} \\ 0 & N_v b\beta_v & N_v b\beta_v & -\mu_v N_v \end{bmatrix}$$

Now, the solutions for the characteristic equation of this last jacobian (eigenvalues of the jacobian) are determined by

$$\lambda = -\mu_h N_h$$

$$\lambda = -\mu_h N_h - N_h r$$

$$\begin{aligned} \mu_h \lambda^2 - (-\mu_h N_h r - \mu_h \mu_v N_v - \mu_h^2 N_h) \lambda + \mu_v N_v N_h \mu_h r \\ - N_v^2 b^2 \beta_v \rho \beta_{ha} - N_v^2 b^2 \beta_v \rho \beta_{hs} + \mu_v N_v N_h \mu_h^2 = 0 \end{aligned}$$

In this point we demand that all the eigenvalues must have negative real part and the condition for it is

$$0 < \mu_v N_v N_h \mu_h r - N_v^2 b^2 \beta_v \rho \beta_{ha} - N_v^2 b^2 \beta_v \rho \beta_{hs} + \mu_v N_v N_h \mu_h^2$$

which can be rewritten as

$$\beta_v < \frac{\mu_v N_h \mu_h (\mu_h + r)}{N_v b^2 \rho (\beta_{ha} + \beta_{hs})}$$

it is to say

$$\frac{\beta_v N_v b^2 \rho (\beta_{ha} + \beta_{hs})}{\mu_v N_h \mu_h (\mu_h + r)} < 1$$

and assuming that the host and vector population have constant size, we can use

$$N_v = \frac{C}{\mu_v}, \rho = \mu_h$$

to obtain

$$\frac{\beta_v C b^2 (\beta_{ha} + \beta_{hs})}{\mu_v^2 N_h (\mu_h + r)} < 1$$

which can be rewritten as $R_0 < 1$, where

$$R_0 = \frac{\beta_v C b^2 (\beta_{ha} + \beta_{hs})}{\mu_v^2 N_h (\mu_h + r)}$$

4. The jacobian for the system (7-10) is evaluated at the endemic equilibrium and for the resulting jacobian the characteristic equation is derived. In order to get that all eigenvalues have negative real parts we demand that

$$0 < b^2 \beta_v \rho N_v \beta_{hs} + b^2 \beta_v \rho N_v \beta_{ha} - \mu_h^2 N_h \mu_v - \mu_h N_h \mu_v r$$

which can be rewritten as

$$\frac{\mu_h N_h \mu_v (\mu_h + r)}{b^2 \rho N_v (\beta_{ha} + \beta_{hs})} < \beta_v$$

it is to say

$$1 < \frac{\beta_v C b^2 (\beta_{ha} + \beta_{hs})}{N_h \mu_v^2 (\mu_h + r)}$$

and for hence

$$1 < R_0$$

The endemic equilibrium can be rewritten as

$$x = \frac{R_0 \mu_v N_h \mu_h + bC \beta_{ha} + bC \beta_{hs}}{(\mu_v N_h \mu_h + bC \beta_{hs} + bC \beta_{ha}) R_0}$$

$$y = \frac{bC\mu_h (R_0 - 1) \beta_{ha}}{(\mu_v N_h \mu_h + bC\beta_{hs} + bC\beta_{ha}) R_0 (\mu_h + r)}$$

$$z = \frac{bC\mu_h (R_0 - 1) \beta_{hs}}{(\mu_v N_h \mu_h + bC\beta_{hs} + bC\beta_{ha}) R_0 (\mu_h + r)}$$

$$w = \frac{\mu_v N_h \mu_h (R_0 - 1)}{R_0 \mu_v N_h \mu_h + bC\beta_{ha} + bC\beta_{hs}}$$

which is indicating that the endemic equilibrium exists when $R_0 > 1$; and then we conclude that the endemic equilibrium is locally stable when exists.

3.2 Mathematical Model with Differential Susceptibility

Theorem 3.2. *For the model with differential susceptibility given by the system (18-22) the following properties hold when host and vector populations with constant size are assumed:*

1. *The free disease equilibrium is given by*

$$x_1 = \frac{\rho_1}{\mu_h} = \frac{\rho_1}{\rho_1 + \rho_2}, x_2 = \frac{\rho_2}{\mu_h} = \frac{\rho_2}{\rho_1 + \rho_2}, y = 0, z = 0, w = 0$$

2. *The free disease equilibrium point is locally stable when $R_0 < 1$ where*

$$R_0 = \frac{Cb^2\beta_v (\rho_2\beta_{hs,2} + \beta_{ha,1}\rho_1 + \beta_{hs,1}\rho_1 + \rho_2\beta_{ha,2})}{\mu_v^2 N_h (\rho_1 + \rho_2) (\rho_1 + \rho_2 + r)}$$

is the basic reproductive number for dengue in the model with two differential susceptibility.

Proof

1. Is directly obtained by computation.
2. The jacobian for the system (18-22) is given by

$$\begin{bmatrix} D & 0 & 0 & 0 & E \\ 0 & F & 0 & 0 & G \\ bwN_v\beta_{ha,1} & bwN_v\beta_{ha,2} & -\mu_h N_h - N_h r & 0 & H \\ bwN_v\beta_{hs,1} & bwN_v\beta_{hs,2} & 0 & -\mu_h N_h - N_h r & I \\ 0 & 0 & J & K & L \end{bmatrix}$$

where D, E, F, G, H, I, J, K and L are given by:

$$D = -bwN_v\beta_{ha,1} - bwN_v\beta_{hs,1} - \mu_h N_h$$

$$E = -bx_1N_v\beta_{ha,1} - bx_1N_v\beta_{hs,1}$$

$$F = -bwN_v\beta_{ha,2} - bwN_v\beta_{hs,2} - \mu_h N_h$$

$$G = -bx_2N_v\beta_{ha,2} - bx_2N_v\beta_{hs,2}$$

$$H = bx_1N_v\beta_{ha,1} + bx_2N_v\beta_{ha,2}$$

$$I = bx_1N_v\beta_{hs,1} + bx_2N_v\beta_{hs,2}$$

$$J = N_v b\beta_v - N_v b\beta_v w$$

$$K = N_v b\beta_v - N_v b\beta_v w$$

$$L = -N_v b\beta_v y - N_v b\beta_v z - \mu_v N_v$$

and when this jacobian is evaluated at the free-disease equilibrium point we obtain

$$\begin{bmatrix} -\mu_h N_h & 0 & 0 & 0 & M \\ 0 & -\mu_h N_h & 0 & 0 & N \\ 0 & 0 & -\mu_h N_h - N_h r & 0 & O \\ 0 & 0 & 0 & -\mu_h N_h - N_h r & P \\ 0 & 0 & N_v b\beta_v & N_v b\beta_v & -\mu_v N_v \end{bmatrix}$$

where M, N, O and P are given by

$$M = -\frac{b\rho_1 N_v \beta_{ha,1}}{\mu_h} - \frac{b\rho_1 N_v \beta_{hs,1}}{\mu_h}$$

$$N = -\frac{b\rho_2 N_v \beta_{ha,2}}{\mu_h} - \frac{b\rho_2 N_v \beta_{hs,2}}{\mu_h}$$

$$O = \frac{b\rho_1 N_v \beta_{ha,1}}{\mu_h} + \frac{b\rho_2 N_v \beta_{ha,2}}{\mu_h}$$

$$P = \frac{b\rho_1 N_v \beta_{hs,1}}{\mu_h} + \frac{b\rho_2 N_v \beta_{hs,2}}{\mu_h}$$

Now, the solutions of the characteristic equation for this last jacobian are given by

$$\lambda = -\mu_h N_h$$

$$\lambda = -\mu_h N_h$$

$$\lambda = -\mu_h N_h - N_h r$$

$$\begin{aligned} 0 = & -\mu_v N_v N_h \mu_h r - \lambda N_h \mu_h r - \mu_v N_v N_h \mu_h^2 - \lambda N_h \mu_h^2 - \mu_h \lambda \mu_v N_v \\ & - \lambda^2 \mu_h + N_v^2 b^2 \beta_v \rho_2 \beta_{ha,2} + N_v^2 b^2 \beta_v \rho_2 \beta_{hs,2} \\ & + N_v^2 b^2 \beta_v \rho_1 \beta_{hs,1} + N_v^2 b^2 \beta_v \rho_1 \beta_{ha,1} \end{aligned}$$

In this point we demand that all the eigenvalues must have negative real part and the condition for it is

$$\begin{aligned} 0 < & \mu_v N_v N_h \mu_h r - N_v^2 b^2 \beta_v \rho_1 \beta_{ha,1} + \mu_v N_v N_h \mu_h^2 - N_v^2 b^2 \beta_v \rho_2 \beta_{hs,2} \\ & - N_v^2 b^2 \beta_v \rho_1 \beta_{hs,1} - N_v^2 b^2 \beta_v \rho_2 \beta_{ha,2} \end{aligned}$$

which can be rewritten as

$$\frac{N_v b^2 \beta_v (\rho_2 \beta_{ha,2} + \rho_1 \beta_{hs,1} + \rho_2 \beta_{hs,2} + \rho_1 \beta_{ha,1})}{\mu_h N_h (\mu_h + r)} < \mu_v$$

and assuming that the host and vector population have constant size, we have that

$$N_v = \frac{C}{\mu_v}, \mu_h = \rho_1 + \rho_2$$

to obtain

$$\frac{Cb^2 \beta_v (\rho_2 \beta_{ha,2} + \rho_1 \beta_{hs,1} + \rho_2 \beta_{hs,2} + \rho_1 \beta_{ha,1})}{\mu_v^2 (\rho_1 + \rho_2) N_h (\rho_1 + \rho_2 + r)} < 1$$

which can be rewritten as $R_0 < 1$, where

$$R_0 = \frac{Cb^2 \beta_v (\rho_2 \beta_{ha,2} + \rho_1 \beta_{hs,1} + \rho_2 \beta_{hs,2} + \rho_1 \beta_{ha,1})}{\mu_v^2 (\rho_1 + \rho_2) N_h (\rho_1 + \rho_2 + r)}$$

Theorem 3.3. *For the model with differential susceptibility with three classes of susceptible individuals in which host and vector populations with constant size are assumed the free-disease equilibrium point is locally stable when $R_0 < 1$, where the basic reproductive number for dengue is now given by*

$$R_0 = \frac{\beta_v b^2 C (\rho_2 \beta_{ha,2} + \rho_1 \beta_{hs,1} + \rho_2 \beta_{hs,2} + \rho_1 \beta_{ha,1} + \rho_3 \beta_{hs,3} + \rho_3 \beta_{ha,3})}{N_h \mu_v^2 (\rho_1 + \rho_2 + \rho_3) (\rho_1 + \rho_2 + \rho_3 + r)}$$

Proof. It is obtained when is introduced a third susceptible group. Explicit proof is not shown here because of space.

Theorem 3.4. *For the model with differential susceptibility with n classes of susceptible individuals in which host and vector populations with constant size are assumed the free-disease equilibrium point is locally stable when $R_0 < 1$, where the basic reproductive number for dengue is now given by*

$$R_0 = \frac{\beta_v b^2 C \sum_{k=1}^n \rho_k \beta_{hs,k} + \rho_k \beta_{ha,k}}{N_h \mu_v^2 \sum_{k=1}^n \rho_k (r + \sum_{k=1}^n \rho_k)}$$

Proof. It is not shown here because of space.

Theorem 3.4. defines a flexible and more accurate approach of the conditions that must be satisfied for having free dengue disease state in populations without homogeneous distributions of biological parameters.

4 Conclusion

In order to develop more accurate models to the real transmission dynamics, it is necessary to identify which assumptions can be taken so the theoretical results don't deviate a lot from evidence results, making the modeling useless and unnecessary for helping epidemiologists to approximate the future course of an outbreak and evaluate strategies to control an epidemic. The basic reproductive number is of vital importance to get this objective and determine when free disease or endemic equilibrium state occurs. So if the considered model is based on firm evidence, it will allow us to find the conditions for having or not those equilibriums. To reduce the deviations produced from coarse assumptions, it is indispensable to take in account those parameters that will lead the spreading dynamics. For dengue disease is important to difference the susceptible populations, because there are factors like heterogeneous mixing of the population due to of cultural and social trends, disease location, rural-urban migration and urban infrastructure that characterize individuals groups with a non-homogeneous biological parameters distribution. Theorems 3.1, 3.2, 3.3 and 3.4 contribute to the control strategies of dengue disease for both homogeneous and heterogeneous distributions of susceptible populations, giving the flexibility and the accurate necessary for an effective analysis of disease spreading of variety systems of individuals if the specific terms for the compartments are known. Taking particular parameters ρ , β_{ha} and β_{hs} for each susceptible compartment contributes to develop a certain and more relevant biological model due to a better modeling of the real system. The results of this paper are also important in order to expand susceptible individuals until n types for having a major adaptation of the model to the biological conditions present where the disease occurs, providing greater flexibility of use to a specific real system.

References

1. World Health Organization, Dengue and Dengue Hemorrhagic Fever, Fact sheet N117 (2009)
2. Pongsumpun, P., Samana, D.: Mathematical Model for Asymptomatic and Symptomatic Infections of Dengue Disease. *WSEAS Transactions on Biology and Biomedicine* 3, 264–269 (2006)
3. Brown, C.W., Kahoui, M.E., Novotni, D., Weber, A.: Algorithmic Methods for Investigating Equilibria in Epidemic Modeling. *Journal of Symbolic Computation* 41, 1157–1173 (2006)
4. Hyman, J.M., Li, J.: Differential Susceptibility Epidemic Models. *Journal of Mathematical Biology* 50(6) (2005)
5. Hincapie, D., Ospina, J., Afuwape, A.U., Ruben, D., Gómez, A.: Epidemic Thresholds in SIR and SIIR Models Applying an Algorithmic Method. *BioSecure* 2008, 119–130 (2008)
6. Taborda, J.A.M.: Epidemic Thresholds via Computer Algebra. In: *MSV 2008*, pp. 178–181 (2008)

Multiple Factorizations of Bivariate Linear Partial Differential Operators

Ekaterina Shemyakova

Research Institute for Symbolic Computation (RISC),
J. Kepler University,
Altenbergerstr. 69, A-4040 Linz, Austria
kath@risc.uni-linz.ac.at
<http://www.risc.uni-linz.ac.at>

Abstract. We study the case when a bivariate Linear Partial Differential Operator (LPDO) of orders three or four has several different factorizations.

We prove that a third-order bivariate LPDO has a first-order left and right factors such that their symbols are co-prime if and only if the operator has a factorization into three factors, the left one of which is exactly the initial left factor, and the right one is exactly the initial right factor. We show that the condition that the symbols of the initial left and right factors are co-prime is essential, and that the analogous statement “as it is” is not true for LPDOs of order four.

Then we consider completely reducible LPDOs, which are defined as an intersection of principal ideals. Such operators may also be required to have several different factorizations. Considering all possible cases, we ruled out some of them from the consideration due to the first result of the paper. The explicit formulae for the sufficient conditions for the complete reducibility of an LPDO were found also.

1 Introduction

The factorization of Linear Partial Differential Operators (LPDOs) is an essential part of recent algorithms for the exact solution for Linear Partial Differential Equations (LPDEs). Examples of such algorithms include numerous generalizations and modifications of the 18th-century Laplace Transformations Method [1,2,3,4,5,6,7,8], the Loewy decomposition method [9,10,11], and others.

The problem of constructing a general factorization algorithm for an LPDO is still an open problem, though several important contributions have been made over the last decades (see for example [9,12,13,14,2,15]). The main difficulty in the case of LPDOs is non-uniqueness of factorization: (irreducible) factors and the number of factors are not necessarily the same for two different factorizations of the same operator. For example, for the famous Landau operator [16] L we have $L = (D_x + 1 + \frac{1}{x+c(y)}) \circ (D_x + 1 - \frac{1}{x+c(y)}) \circ (D_x + xD_y) = (D_{xx} + xD_{xy} + D_x + (2+x)D_y) \circ (D_x + 1)$. Note that the second order factor in the second factorization is hyperbolic and is irreducible.

However, for some classes of LPDOs factorization is unique. For example, there is [9] no more than one factorization that extends a factorization of the principal symbol of the operator into co-prime factors (see Theorem [1]).

Some important methods of exact integration, for example, mentioned above Loewy decomposition methods require LPDOs to have a number of different factorizations of certain types. Also completely reducible LPDOs introduced in [9], which becomes significant as the solution space of a completely reducible LPDO coincides with the sum of those of its irreducible right factors may require a number of right factors. Thus, in Sec. [4] we study the case when a bivariate (not necessarily hyperbolic) LPDO has two different factorizations. For operators of order three we have a really interesting result (Theorems [4] and [5]). We showed that analogous statement for operators of order four is not true.

For the proof of the theorems we use invariants' methods. Invariants of LPDOs under the gauge transformations (see Sec. [2]) are widely used for factorization problems since Laplace' times as many properties appearing in connection with the factorization of an LPDO are invariant under the gauge transformations, and, therefore, can be expressed in terms of generating invariants, which were found in [17]. Factorization itself is invariant under the gauge transformations: if for some LPDO L , $L = L_1 \circ L_2$, then $L^g = L_1^g \circ L_2^g$. Expressions for necessary and sufficient conditions for the existence of a factorization of a given LPDO of a given factorization type were found in [18] and [19]. We use these expressions in the proofs of the theorems of Sec. [5].

Theorems [4] and [5] of Section [4] allow us to reduce consideration of cases in Sec. [5], where we show how the problem of the complete reducibility of a hyperbolic bivariate LPDO can be expressed in terms of invariants also.

2 Definitions and Notations

Consider a field K of characteristic zero with commuting derivations ∂_x, ∂_y , and the ring of linear differential operators $K[D] = K[D_x, D_y]$, where D_x, D_y correspond to the derivations ∂_x, ∂_y , respectively. In $K[D]$ the variables D_x, D_y commute with each other, but not with elements of K . For $a \in K$ we have $D_i a = a D_i + \partial_i(a)$. Any operator $L \in K[D]$ has the form $L = \sum_{i+j=0}^d a_{ij} D_x^i D_y^j$, where $a_{ij} \in K$. The polynomial $\text{Sym}(L) = \sum_{i+j=d} a_{ij} X^i Y^j$ in formal variables X, Y is called the (principal) *symbol* of L .

Below we assume that the field K is differentially closed unless stated otherwise, that is it contains solutions of (non-linear in the generic case) differential equations with coefficients from K .

Let K^* denote the set of invertible elements in K . For $L \in K[D]$ and every $g \in K^*$ consider the gauge transformation $L \rightarrow L^g = g^{-1} \circ L \circ g$. Then an algebraic differential expression I in the coefficients of L is *invariant* under the gauge transformations (we consider only these in the present paper) if it is unaltered by these transformations. Trivial examples of invariants are the coefficients of the symbol of an operator. A generating set of invariants is a set using which all possible differential invariants can be expressed.

Given a third-order bivariate LPDO L and a factorization of its symbol $\text{Sym}(L)$ into first-order factors. In some system of coordinates the operator has one of the following normalized forms:

$$L = (p(x, y)D_x + q(x, y)D_y)D_xD_y + \sum_{i+j=0}^2 a_{ij}(x, y)D_x^iD_y^j, \tag{1}$$

$$L = D_x^2D_y + \sum_{i+j=0}^2 a_{ij}(x, y)D_x^iD_y^j, \tag{2}$$

$$L = D_x^3 + \sum_{i+j=0}^2 a_{ij}(x, y)D_x^iD_y^j, \tag{3}$$

where $p = p(x, y) \neq 0$, $q = q(x, y) \neq 0$, $a_{ij} = a_{ij}(x, y)$. The normalized form **(1)** has symbol $S = pX + qY$. Without loss of generality one can assume $p = 1$. Each of the class of operators admits gauge transformations $L \rightarrow g^{-1} \circ L \circ g$ for $g = g(x, y) \neq 0$.

Remark 1. Recall that since for two LPDOs $L_1, L_2 \in K[D]$ we have $\text{Sym}(L_1 \circ L_2) = \text{Sym}(L_1) \cdot \text{Sym}(L_2)$, any factorization of an LPDO extends some factorization of its symbol. In general, if $L \in K[D]$ and $\text{Sym}(L) = S_1 \cdot \dots \cdot S_k$, then we say that the factorization

$$L = F_1 \circ \dots \circ F_k, \quad \text{Sym}(F_i) = S_i, \quad \forall i \in \{1, \dots, k\},$$

is of the *factorization type* $(S_1) \dots (S_k)$.

We reformulate the famous result of **[9]** in the new notation:

Theorem 1. **[9]** *Let LPDO L of arbitrary order and in arbitrary number of independent variables have symbol $\text{Sym}(L) = S_1 \dots S_k$, where S_i -s are pairwise coprime. Then there exists at most one factorization of L of the type $(S_1) \dots (S_k)$.*

3 Factorization via Invariants for Hyperbolic Bivariate Operators of Order Three

Information from this section will be used in the proofs below in the case, where L is hyperbolic operator.

Theorem 2. **[20]** *The following 7 invariants form a generating set of invariants for operators of the form **(1)**: $q, I_1 = 2q^2a_{20} - qa_{11} + 2a_{02}, I_2 = -qa_{02y} + a_{02}q_y + q^2a_{20x}, I_3 = a_{10} + 2q_ya_{20} + a_{20}^2q - a_{11y} + qa_{20y} - a_{11}a_{20}, I_4 = a_{01}q^2 - 3q_xa_{02} + a_{02}^2 - a_{11x}q^2 + a_{11}qq_x + qa_{02x} - a_{02}a_{11}q, I_5 = a_{00}q + 2a_{02}a_{20x} - a_{02}a_{10} - a_{01}a_{20}q - \frac{1}{2}a_{11xy}q + qq_xa_{20y} - a_{11}qa_{20x} + qq_ya_{20x} + 2q^2a_{20}a_{20x} + qq_xy a_{20} + a_{20}a_{11}a_{02}$.*

The set of values of these seven invariants uniquely defines an equivalent class of operators of the form **(1)**. Also invariant properties of such operators can be described in terms of the seven invariants.

Lemma 1. *The property of having a factorization (or a factorization extending a certain factorization of the symbol) is invariant.*

Proof. Let $L = F_1 \circ F_2 \circ \dots \circ F_k$, for some operators $F_i \in K[D]$. For every $g \in K^*$ we have $g^{-1} \circ L \circ g = (g^{-1} \circ F_1 \circ g) \circ (g^{-1} \circ F_2 \circ g) \circ \dots \circ (g^{-1} \circ F_k \circ g)$.

Looking through the formulae of the next theorem, notice that some conditions are the same for different types of factorizations. In particular, one can pay attention to conditions $(A_1) - (D_1)$. Such correlations will be used in the next section (Sec. 4).

Theorem 3. [18] *Given the values of the invariants $q, I_1, I_2, I_3, I_4, I_5$ (from Theorem 2) for an equivalence class of operators of the form (1). The LPDOs of the class have a factorization of factorization type*

$(S)(XY)$ if and only if

$$\left. \begin{aligned} I_3q^3 - I_{1y}q^2 + q_yI_1q - I_4 + qI_{1x} - 2q_xI_1 - 3qI_2 &= 0, \\ -q^2I_{4y} + 1/2q^3I_{1xy} - qI_{4x} - 3/2q^2q_xI_{1y} + q^3I_5 + q^2I_{1xx} \\ - 3/2I_1q^2q_{xy} - 2I_1qq_{xx} + 5I_1qq_xq_y + 6I_1q_x^2 + 3I_4q_x \\ + 3I_4qq_y - qI_1I_{1x} + I_1I_4 + 2q_xI_1^2 - 4I_{1x}qq_x - 3/2I_{1x}q^2q_y \\ - 2q^2I_{2x} - q^3I_{2y} + I_2qI_1 + 4I_2qq_x + 2I_2q^2q_y &= 0; \end{aligned} \right\} \quad (4)$$

$(S)(X)(Y)$ if and only if (4) & $-I_4 + qI_{1x} - 2q_xI_1 - qI_2 = 0$;

$(S)(Y)(X)$ if and only if (4) & (C_1) ;

$(X)(SY)$ if and only if

$$\left. \begin{aligned} (D_1) : \quad qq_{xx} - I_4 - 2q_x &= 0, \\ -3/2q_xqI_{1y} - q^3I_{3x} + I_5q^2 + 1/2q^2I_{1xy} - 1/2qq_yI_{1x} + \\ q_xq^2I_3 + 2I_1q_xq_y - 1/2I_1q_{xy}q - 4q_xI_2 + qI_{2x} &= 0. \end{aligned} \right\} \quad (5)$$

$(X)(S)(Y)$ if and only if (5) & (B_1) ;

$(X)(Y)(S)$ if and only if (5) & (D_1) ;

$(XY)(S)$ if and only if

$$\left. \begin{aligned} -qI_2 + qq_xq_y + q_{yy}q^3 - q^2q_{xy} + qq_{xx} + I_3q^3 - I_4 - 2q_x^2 &= 0, \\ q^3I_5 + qI_{4x} + 1/2q^3I_{1xy} - 3/2q^2q_xI_{1y} + I_1I_4 + q^2I_{2x} + 2I_1qq_xq_y \\ + 2I_1q_x^2 - 5I_4q_x - 1/2I_1q^2q_{xy} - I_1qq_{xx} + I_4qq_y - 1/2I_{1x}q^2q_y - 4I_2qq_x \\ - 10q_x^3 - q^2q_{xxx} - q^4I_{3x} + I_3q^3q_x + 2qq_x^2q_y - q^2q_yq_{xx} + 8qq_xq_{xx} &= 0. \end{aligned} \right\} \quad (6)$$

$(YS)(X)$ if and only if

$$\left. \begin{aligned} (C_1) : \quad -2q_xI_1 + qI_{1x} - I_4 - 2qI_2 &= 0, \\ -qI_{4y} + 1/2q^2I_{1xy} + I_5q^2 - I_2I_1 - q^2I_{2y} + 2q_yI_4 + 3I_1q_xq_y - \\ 3/2I_1q_{xy}q - 1/2qq_yI_{1x} - 3/2q_xqI_{1y} &= 0. \end{aligned} \right\} \quad (7)$$

$(XS)(Y)$ if and only if

$$\left. \begin{aligned} (B_1) : \quad I_3q^2 - qI_{1y} + q_yI_1 - 2I_2 &= 0 \\ -1/2qq_yI_{1x} - 3/2q_xqI_{1y} + I_5q^2 - q^3I_{3x} + 1/2q^2I_{1xy} + q_xq^2I_3 \\ - 2q_xI_2 + qI_{2x} + 3I_1q_xq_y - 3/2I_1q_{xy}q - I_2I_1 + 2q_y^2q_xq - 2q_yqI_2 \\ - 2q_{xy}q^2q_y + 2q_{xy}qq_x - 2q_yq_x^2 &. \end{aligned} \right\} \quad (8)$$

$(Y)(SX)$ if and only if

$$(A_1) : \left. \begin{aligned} I_3 + q_{yy} &= 0, \\ -qI_{4y} - 3/2q_xqI_{1y} + I_5q^2 + 1/2q^2I_{1xy} + 2q_yI_4 - 1/2qq_yI_{1x} + \\ 3I_1q_xq_y - 3/2I_1q_{xy}q - q^2I_{2y} &= 0; \end{aligned} \right\} \quad (9)$$

$(Y)(X)(S)$ if and only if (6) & $-qq_{xx} + I_4 + 2q_x^2 + qI_2 - qq_xq_y + q^2q_{xy} = 0$;
 $(Y)(S)(X)$ if and only if (7) & (A_1) ;

For LPDOs of the forms (2) and (3) generating sets of invariants and the corresponding conditions of the existence of factorizations of different types are known also [17], [19].

4 Several Factorizations of One Operator

Theorem 4. Let $\gcd(S_1, S_2) = 1$. A third-order bivariate operator L has a first-order left factor of the symbol S_1 and a first-order right factor of the symbol S_2 if and only if it has a complete factorization of the type $(S_1)(T)(S_2)$, where $T = \text{Sym}(L)/(S_1S_2)$.

The following diagram is an informal illustration of the statement of the theorem:

$$((S_1)(\dots) \wedge (\dots)(S_2)) \iff (S_1)(\dots)(S_2)$$

Proof. The part of the statement “ \Leftarrow ” is trivial. Prove “ \Rightarrow ”.

The symbol of the operator has two different factors, therefore, the normalized form of the operator L is either (1) or (2). Without loss of generality we can consider L in its normalized form.

Consider the first (hyperbolic) case. For this class of operators we have the generating system of invariants $q, I_1, I_2, I_3, I_4, I_5$ from Theorem 2. The symbol is the same for all the operators in the class and is $X \cdot Y \cdot S$, where $S = pX + qY$. The following six cases are the only possibilities for the S_1 and S_2 , $\gcd(S_1, S_2) = 1$.

Case $S_1 = S, S_2 = Y$. By the Theorem 3 operator L has a left factor of the symbol $S_1 = S$ and a right factor of the symbol $S_2 = Y$ if and only if conditions (4) and (8) are satisfied. From the second equality in (8) derive an expression for I_3 and substitute it for I_3 into the first equality in (4). The resulting equality implies that the third condition (in Theorem 3) for the existence of factorization of the type $(S)(X)(Y)$ is satisfied. The remaining two first conditions are exactly the same as two conditions (4), and, therefore, the theorem is proved for this case.

Case $S_1 = Y, S_2 = S$. Operator L has a left factor of the symbol $S_1 = Y$ and a right factor of the symbol $S_2 = S$ if and only if conditions (9) and (6) are satisfied. From the first equality in (9) derive an expression for I_3 ($I_3 = -q_{yy}$) and substitute it for I_3 into the first equality in (6). The resulting equality implies that the third condition (in Theorem 3) for the existence of factorization of the type $(Y)(X)(S)$ is satisfied. Since the first two conditions are satisfied obviously, we proved the theorem for this case.

Cases $S_1 = X, S_2 = Y$ and $S_1 = Y, S_2 = X$ and $S_1 = X, S_2 = S$ and $S_1 = S, S_2 = X$ are obvious consequences of Theorem 3.

Consider the case, where the symbol of L has exactly two different factors, that is L is in the normalized form (2). There are only two cases to consider: $S_1 = X, S_2 = Y$, and $S_1 = Y, S_2 = X$.

Straightforward computations show that the equality $H_1 \circ H_2 = G_2 \circ G_1$, where $\text{Sym}(H_1) = X$ and $\text{Sym}(G_1) = Y$ implies that for some $a = a(x, y)$ and $b = b(x, y)$ we have $H_2 = D_{xy} + aD_x + bD_y + a_x + ab = (D_x + b) \circ (D_y + a)$, while $G_1 = D_y + a$. Thus, H_2 has a factorization of the type $(X)(Y)$.

Similar computations show that the equality $H_1 \circ H_2 = G_2 \circ G_1$, where $\text{Sym}(H_1) = Y$ and $\text{Sym}(G_1) = X$ implies that for some $a = a(x, y)$ and $b = b(x, y)$ we have $G_2 = D_{xy} + aD_x + bD_y + b_y + ab = (D_y + a) \circ (D_x + b)$, while $H_1 = D_y + a$. Thus, H_2 has a factorization of the type $(Y)(X)$.

Example 1 (Symbol $SXY, S_1 = X + Y, S_2 = Y$). We found an operator with two factorizations $L = (D_x + D_y + x) \circ (D_{xy} + yD_x + y^2D_y + y^3)$ and $L = (D_{xx} + D_{xy} + (x + y^2)D_x + y^2D_y + xy^2 + 2y) \circ (D_y + y)$. Then L has factorization $L = (D_x + D_y + x) \circ (D_x + y^2) \circ (D_y + y)$.

Example 2 (Symbol $X^2Y, S_1 = X, S_2 = Y$). We found an operator with two factorizations $L = (D_x + x) \circ (D_{xy} + yD_x + y^2D_y + y^3)$ and $L = (D_{xx} + (x + y^2)D_x + xy^2) \circ (D_y + y)$. Then L has factorization $L = (D_x + x) \circ (D_x + y^2) \circ (D_y + y)$.

Example 3 (Symbol $X^2Y, S_1 = Y, S_2 = X$). We found an operator with two factorizations $L = (D_y + x) \circ (D_{xx} + yD_x + y^3 - y^4)$ and $L = (D_{xy} + xD_x + y^2D_y + xy^2 + 2y) \circ (D_x + y - y^2)$. Then L has factorization $L = (D_y + x) \circ (D_x + y^2) \circ (D_x + y - y^2)$.

Looking at the examples, one can notice that the factorizations into first-order factors have the right and left factors exactly the same as they were in the initial, given factorizations. In fact, this will be always the case. Accordingly, we improve Theorem 4 proving the following one.

Theorem 5. *A third-order bivariate operator L has a first-order left factor F_1 and a first-order right factor F_2 with $\text{gcd}(\text{Sym}(F_1), \text{Sym}(F_2)) = 1$ if and only if L has a factorization into three factors, the left one of which is exactly F_1 and the right one is exactly F_2 .*

The following diagram is an informal illustration of the statement of the theorem:

$$(L = F_1 \circ \dots \quad \wedge \quad L = \dots \circ F_2) \iff L = F_1 \circ \dots \circ F_2 .$$

Proof. Let L have the normalized form (1). Then by Theorem 4 if L has a first-order left factor F_1 and a first-order right factor F_2 ($\text{Sym}(F_2)$ is co-prime with $\text{Sym}(F_1)$), it has a factorization into first-order factors of the type $(S_1)(R)(S_2)$, where $R = \text{Sym}_L / (S_1S_2)$. Theorem 1 implies that such factorization is unique, so we have some unique first-order LPDOs T_1, T, T_2 such that $L = T_1 \circ T \circ T_2$, where $\text{Sym}(T_1) = S_1, \text{Sym}(T) = R, \text{Sym}(T_2) = S_2$. This also means that there

are factorization $L = T_1 \circ (T \circ T_2)$ of the type $(S_1)(RS_2)$ and factorization $L = (T_1 \circ T) \circ T_2$ of the type $(S_1R)(S_2)$. Since S_1, R, S_2 are pairwise coprime, by Theorem 1 such factorizations are unique. On the other hand we have initial factorizations that are factorizations of the same types. Thus, we have $F_1 = T_1$ and $F_2 = T_2$.

For L that has the normalized form (2), the statement of the theorem is actually a subresult in the proof of Theorem 4 for this case.

Proposition 1. *The condition $\gcd(S_1, S_2) = 1$ in Theorems 4 and 5 cannot be omitted.*

Proof. Hyperbolic case. Consider an equivalence class of (1) defined by $q = 1, I_1 = I_2 = I_5 = 0, I_3 = I_4 = x - y$ of the invariants from Theorem 2. Using Theorem 3 one can verify that operators of the class have factorizations of the types $(S)(XY)$ and $(XY)(S)$ only.

Such equivalence class is not empty. For example, operator $A_3 = D_{xxy} + D_{xyy} + (x - y)(D_x + D_y)$ belongs to this equivalence class. Only the following two factorizations exist for A_3 : $A_3 = (D_{xy} + x - y)(D_x + D_y) = (D_x + D_y)(D_{xy} + x - y)$.

The non-hyperbolic case. Consider operator of Landau

$$D_x^3 + xD_x^2D_y + 2D_x^2 + (2x + 2)D_xD_y + D_x + (2 + x)D_y ,$$

which has two factorizations into different numbers of irreducible factors:

$$L = Q \circ Q \circ P = R \circ Q ,$$

for the operators $P = D_x + xD_y, Q = D_x + 1, R = D_{xx} + xD_{xy} + D_x + (2 + x)D_y$. That is factorizations of the types $(X)(SX), (SX)(X)$ exist, while those of the type $(X)(S)(X)$ do not. Here we denote $S = X + xY$.

Proposition 2. *The statement of Theorem 5 is not always true for a general fourth-order hyperbolic operator.*

Proof. For example, operator

$$\begin{aligned} L &= (D_x + D_y) \circ (D_xD_y(D_x + D_y) + xD_{xx} + (2 - x^2)D_x + xD_y - 2x + x^2) \\ &= (D_x(D_x + D_y)^2 - xD_x(D_x + D_y) + (x - 2)D_x + (x - 1)D_y + 1) \circ (D_y + x) . \end{aligned}$$

The second factor in the first factorization has no factorization.

5 Completely Reducible Operators

Let $\langle L \rangle$ denote the left ideal generated by an operator $L \in K[D]$. Consider Linear Ordinary Differential Operators (LODOs). The ring of LODOs are the principal ideal domain and, therefore, the intersection of two principal ideals is again principle. Consequently, the least common multiple (lcm) of two LODOs L_1 and L_2 can be defined uniquely as L such that $\langle L \rangle = \langle L_1 \rangle \cap \langle L_2 \rangle$. Since in the ring of LPDOs this is not the case, it was suggested [9] to introduce the notion of a completely irreducible LPDO.

Definition 1. [9] An LPDO L is said to be completely irreducible, if it can be expressed as $\langle L \rangle = \langle L_1 \rangle \cap \dots \cap \langle L_k \rangle$ for suitable irreducible LPDOs L_1, \dots, L_k . In this case $L = \text{lcm}\{L_1, \dots, L_k\}$ by definition.

Theorem 6. [9] If an LPDO L has right factors L_1, \dots, L_k and

$$\text{Sym}_L = \text{lcm}(\text{Sym}_{L_1}, \dots, \text{Sym}_{L_k}) , \tag{10}$$

then $\langle L \rangle = \langle L_1 \rangle \cap \dots \cap \langle L_k \rangle$. If the factors L_1, \dots, L_k are irreducible, then L is completely reducible via L_1, \dots, L_k .

An additional piece of motivation is [9] the following. Let for an ideal $I \subset K[D]$ denote by $V_I \subset K$ its space of solutions. Then for two ideals $I_1, I_2 \subset K[D]$ we have [21][22] $V_{I_1 \cap I_2} = V_{I_1} + V_{I_2}$, which allows to reduce the solution problem of the partial differential equation corresponding to a completely reducible LPDO to ones of corresponding to its factors.

Notice that the properties of the existence of a right factor with certain symbol or a factorization of certain factorization type, and, therefore, irreducibility of factors are invariant under the gauge transformations. Consequently, an invariant description of the completely reducible operators is possible.

Consider a hyperbolic linear partial differential operator of third order in the normalized form (II). Consider all possible right factors of the operator, their symbols are

$$X, Y, S, XY, XS, YS,$$

where $S = X + qY$. Let us list all the possibilities for $\{L_1, \dots, L_k\}$ that L_i -s together satisfy (II). Notice that the number (of factors) k is not fixed. However, by Theorem I there is no more than one factorization of each factorization type. Thus, $L_i \neq L_j$ for $i \neq j$, and, therefore, $k \leq 6$.

- I. $\{X, Y, S\}$;
- II. $\{SX, SY\}$, $\{SX, XY\}$, $\{SY, XY\}$;
- III. $\{X, SY\}$, $\{Y, SX\}$, $\{S, XY\}$ – the right factors are co-prime ;
- IV. $\{X, Y, SX\}$, $\{X, XY, SY\}$, ... – the sets that contain as the subset one of the sets of the groups III and I.
- V. $\{SX, SY, XY\}$ – the only set that contains as the subset one of the sets of the groups II and does not belong to the group IV.

Theorem 4 allows us to avoid the consideration of the large group IV. Indeed, by Theorem 4, at least one of the second-order factors of the sets fails to be irreducible.

Now, when we rule out all cases but eight, using Theorem 3 it is easy to obtain sufficient conditions for LPDOs to be completely reducible with

$$\langle L \rangle = \langle L_1 \rangle \cap \dots \cap \langle L_k \rangle ,$$

where $\{\text{Sym}(L_1), \dots, \text{Sym}(L_k)\}$ belong to I or II or III or V. We just combine certain conditions from Theorem 3. Further below we use notation \overline{W} for an arbitrary operator with the principal symbol W .

It is of interest to consider instead an important particular case $q = 1$. We collect the sufficient conditions for this case in the following theorem.

Theorem 7. *Given an equivalence class of (1) by $q = 1$, and values I_1, \dots, I_5 of the invariants from Theorem 2. Operators of the class are completely reducible with*

$$\begin{aligned}
 I. \quad \langle L \rangle &= \langle \overline{X} \rangle \cap \langle \overline{Y} \rangle \cap \langle \overline{S} \rangle \text{ if} \\
 & (D_y + I_1) \circ (2D_x + D_y)(I_1) = 0, \\
 & I_2 = I_{1x} - I_3, \\
 & I_3 = (I_{1y} + 2I_{1x})/3, \\
 & I_4 = -I_2 + I_3, \\
 & I_5 = I_1 I_{1x} - 2I_1 I_3 - I_{1xy}/2;
 \end{aligned}$$

$$\begin{aligned}
 II. \quad \langle L \rangle &= \langle \overline{SX} \rangle \cap \langle \overline{SY} \rangle \text{ if} \\
 & I_2 = F_1(y - x), \\
 & I_3 = I_4 = 0, \\
 & I_5 = -1/2 I_{1xy} + I_{2y},
 \end{aligned}$$

where $F_1(y - x)$ is some function;

$$\begin{aligned}
 \langle L \rangle &= \langle \overline{SX} \rangle \cap \langle \overline{XY} \rangle \text{ if} \\
 & I_{1xy} + I_{2x} - I_1 I_{1y} - 2I_1 I_2 = 0, \\
 & I_3 = 0, \\
 & I_4 = -I_{1y} + I_{1x} - 3I_2, \\
 & I_5 = I_{4y} - I_{1xy}/2 + I_{2y};
 \end{aligned}$$

$$\begin{aligned}
 \langle L \rangle &= \langle \overline{SY} \rangle \cap \langle \overline{XY} \rangle \text{ if} \\
 & I_{1xy} - I_1 I_{1x} - I_{2y} + I_1 I_2 = 0, \\
 & I_3 = I_{1y} - I_{1x} + 3I_2, \\
 & I_4 = 0, \\
 & I_5 = I_{3x} - 1/2 I_{1xy} - I_{2x};
 \end{aligned}$$

$$\begin{aligned}
 III. \quad \langle L \rangle &= \langle \overline{X} \rangle \cap \langle \overline{SY} \rangle \text{ if} \\
 & -I_{3x} + (I_{1x} I_1 + I_{1xy} + I_{1xx})/2 = 0, \\
 & I_2 = I_{1x}/2, \\
 & I_4 = 0, \\
 & I_5 = -1/2 I_{1xy} + I_2 I_1 + I_{2y};
 \end{aligned}$$

$$\begin{aligned}
 \langle L \rangle &= \langle \overline{Y} \rangle \cap \langle \overline{SX} \rangle \text{ if} \\
 & -I_{4y} - I_{1y} I_1/2 + I_{1xy}/2 + I_{1yy}/2 = 0, \\
 & I_2 = -I_{1y}/2, \\
 & I_3 = 0, \\
 & I_5 = I_2 I_1 - I_{1xy}/2 - I_{2x};
 \end{aligned}$$

$$\begin{aligned}
 \langle L \rangle = \langle \overline{S} \rangle \cap \langle \overline{XY} \rangle \text{ if} \\
 -I_{3y} + (I_{1xy} + I_{1xx} - I_1 I_{1y} - I_1 I_{1x})/2 - I_{3x} = 0, \\
 I_2 = (I_{1x} - I_{1y})/2, \\
 I_4 = -I_2 + I_3, \\
 I_5 = (I_1 I_{1x} - I_1 I_{1y} - I_{1xy})/2 - I_1 I_3;
 \end{aligned}$$

$$\begin{aligned}
 V. \langle L \rangle = \langle \overline{SX} \rangle \cap \langle \overline{SY} \rangle \cap \langle \overline{XY} \rangle \text{ if} \\
 I_{1xx} - I_{1yy} = 0, \\
 I_{1xx} - 2I_1 I_{1x} + 2I_{1xy} - I_1 I_{1y} = 0, \\
 I_2 = (I_{1x} - I_{1y})/3, \\
 I_3 = I_4 = 0, \\
 I_5 = -I_{1xy}/2 + I_{2y}.
 \end{aligned}$$

6 Conclusions

The paper is devoted to the case when one LPDO has several factorizations.

In Sec. 4 we proved that a third-order bivariate operator L has a first-order left factor F_1 and a first-order right factor F_2 with $\gcd(\text{Sym}(F_1), \text{Sym}(F_2)) = 1$ if and only if L has a factorization into three factors, the left one of which is exactly F_1 and the right one is exactly F_2 . Also it was shown that the condition $\gcd(\text{Sym}(F_1), \text{Sym}(F_2)) = 1$ is essential, and that the analogous statement “as it is” is not true for LPDOs of order four. However, other generalizations may be possible.

The proof for the hyperbolic case was done using invariants’ methods. This is a nice and easy way to prove the things since the expressions for the necessary and sufficient conditions of the existence of factorizations of a given type are already known. It was the form of the conditions that allowed us to make initial hypotheses that were proved later to be true in Sec. 4. However, some other method is required for generalizations to higher order LPDOs.

In Sec. 5 we considered the case, where one LPDO has two or more several factorizations of certain types. Most of the cases were ruled out from the consideration due to the results of Sec. 4. The explicit formulae for the sufficient conditions for the complete reducibility of an LPDO were found for the case $p = 1$ (which is the case where the symbol of L has constant coefficients only).

Acknowledgments. The author was supported by the Austrian Science Fund (FWF) under project DIFFOP, Nr. P20336-N18.

References

1. Tsarev, S., Shemyakova, E.: Differential transformations of parabolic second-order operators in the plane. In: Proc. Steklov Inst. Math., Moscow (2009), <http://arxiv.org/abs/0811.1492>
2. Tsarev, S.: Generalized laplace transformations and integration of hyperbolic systems of linear partial differential equations. In: ISSAC 2005: Proc. 2005 Int. Symp. on Symbolic and Algebraic Computation, pp. 325–331. ACM Press, New York (2005)

3. Tsarev, S.: Factorization of linear partial differential operators and darboux' method for integrating nonlinear partial differential equations. *Theo. Math. Phys.* 122, 121–133 (2000)
4. Anderson, I., Juras, M.: Generalized Laplace invariants and the method of Darboux. *Duke J. Math.* 89, 351–375 (1997)
5. Anderson, I., Kamran, N.: The variational bicomplex for hyperbolic second-order scalar partial differential equations in the plane. *Duke J. Math.* 87, 265–319 (1997)
6. Athorne, C.: A $z \times r$ toda system. *Phys. Lett. A.* 206, 162–166 (1995)
7. Zhiber, A.V., Startsev, S.Y.: Integrals, solutions and existence of the laplace transformations for a linear hyperbolic system of equations. *Math. Notes* 74(6), 848–857 (2003)
8. Startsev, S.: Cascade method of laplace integration for linear hyperbolic systems of equations. *Mathematical Notes* 83 (2008)
9. Grigoriev, D., Schwarz, F.: Factoring and solving linear partial differential equations. *Computing* 73(2), 179–197 (2004)
10. Grigoriev, D., Schwarz, F.: Generalized loewy-decomposition of d-modules. In: *ISSAC 2005: Proc. 2005 Int. Symp. on Symbolic and Algebraic Computation*, pp. 163–170. ACM, New York (2005)
11. Grigoriev, D., Schwarz, F.: Loewy decomposition of third-order linear pde's in the plane. In: *ISSAC 2008: Proc. 2005 Int. Symp. on Symbolic and Algebraic Computation*, pp. 277–286. ACM, New York (2008)
12. Li, Z., Schwarz, F., Tsarev, S.P.: Factoring systems of linear pdes with finite-dimensional solution spaces. *J. Symb. Comput.* 36(3-4), 443–471 (2003)
13. Li, Z., Schwarz, F., Tsarev, S.: Factoring zero-dimensional ideals of linear partial differential operators. In: *ISSAC 2002: Proc. 2002 Int. Symp. on Symbolic and Algebraic Computation*, pp. 168–175. ACM Press, New York (2002)
14. Tsarev, S.P.: An algorithm for complete enumeration of all factorizations of a linear ordinary differential operator. In: *ISSAC 1996: Proc. 1996 Int. Symp. on Symbolic and Algebraic Computation*, pp. 226–231. ACM, New York (1996)
15. Shemyakova, E., Winkler, F.: Obstacles to the Factorization of Linear Partial Differential Operators into Several Factors. *Programming and Computer Software* 33(2), 67–73 (2007)
16. Blumberg, H.: Über algebraische Eigenschaften von linearen homogenen Differentialausdrücken. PhD thesis, Göttingen (1912)
17. Shemyakova, E., Mansfield, E.: Moving frames for laplace invariants. In: *Proc. ISSAC 2008 The International Symposium on Symbolic and Algebraic Computation*, pp. 295–302 (2008)
18. Shemyakova, E., Winkler, F.: On the invariant properties of hyperbolic bivariate third-order linear partial differential operators. In: *Kapur, D. (ed.) ASCM 2007. LNCS (LNAI), vol. 5081*, pp. 199–212. Springer, Heidelberg (2008)
19. Shemyakova, E.: On the invariant properties of non-hyperbolic third-order linear partial differential operators. In: *Conferences on Intelligent Computer Mathematics*, vol. 5625 (2009)
20. Shemyakova, E., Winkler, F.: A full system of invariants for third-order linear partial differential operators in general form. In: *Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2007. LNCS, vol. 4770*, pp. 360–369. Springer, Heidelberg (2007)
21. Cassidy, P.: Differential algebraic groups. *Amer. J. Math.* 94, 891–895 (1972)
22. Sit, W.: Typical differential dimension of the intersection of linear differential algebraic groups. *J. Algebra* 32(3), 476–487 (1974)

Computing Gröbner Bases within Linear Algebra

Akira Suzuki*

Kobe University
Rokkodai-Cho 1-1, Nada, Kobe, Japan
sakira@kobe-u.ac.jp

Abstract. In this paper, we present an alternative algorithm to compute Gröbner bases, which is based on computations on sparse linear algebra. Both of S-polynomial computations and monomial reductions are computed in linear algebra simultaneously in this algorithm. So it can be implemented to any computational system which can handle linear algebra. For a given ideal in a polynomial ring, it calculates a Gröbner basis along with the corresponding term order appropriately.

1 Introduction

The concept of Gröbner bases and the algorithm to compute them was introduced by Buchberger [3] and the algorithm consists of computation of S-polynomials and one of monomial reductions. Since it has applications across a wide range of computer algebra, many optimizations have been studied and developed [1, 2, 9, 10, 11, 14]. In [6, 7], Faugère introduced a new efficient algorithm F_4 and F_5 to compute Gröbner bases by use of linear algebra in order to achieve simultaneous monomial reductions. A method to solve systems of algebraic equations by use of linear algebra were also studied by Lazard in [12]. Also by appropriate choices of term orders for given ideals, we may compute Gröbner basis for it efficiently. Mora-Robbiano [13] and Caboara [4] studied several method to find suitable term orders.

In this paper, we introduce a new method to compute Gröbner bases of ideals on polynomial rings by use of sparse linear algebra. It implicitly processes both of the computations of S-polynomials and the one of monomial reductions in a single linear space simultaneously in the form of Gaussian elimination. Thus, with this method, we can expect to use a great variety of techniques including parallelism used by computation in linear algebras, in order to get Gröbner bases. Moreover our algorithm includes an indicator to choose an appropriate term order for an efficient computation of Gröbner basis for a given system of polynomials. The term order changes dynamically during the computation.

Let K be a field. Let $\bar{X} = \{X_1, \dots, X_n\}$ be a finite set of variables, i.e., n is the number of invariants. Let $T(\bar{X})$ be the set of the terms of \bar{X} . Throughout this

* This work was supported by KAKENHI (20500013).

paper, we use roman letters a, b, c, \dots for elements of K , Greek letters $\alpha, \beta, \gamma, \dots$ for terms in $T(\bar{X})$, f, g, h, \dots for polynomials in $K[\bar{X}]$.

For a term order \prec on $T(\bar{X})$ and a polynomial $f \in K[\bar{X}]$, we let $\text{terms}(f) \subseteq T(\bar{X})$ be the finite set of the terms appearing in f , and $\text{ht}_\prec(f) \in T(\bar{X})$ be the \prec -maximal term of f , i.e., $\text{ht}_\prec(f) \in \text{terms}(f)$ and $\alpha \prec \text{ht}_\prec(f)$ for all $\alpha \in \text{terms}(f) \setminus \{\text{ht}_\prec(f)\}$. We also let $\text{hc}_\prec(f) \in K$ be the corresponding coefficient of $\text{ht}_\prec(f)$ in f , and let $\text{hm}_\prec(f) = \text{hc}_\prec(f) \cdot \text{ht}_\prec(f)$.

For $f \in K[\bar{X}]$, we denote the partial degree of f with respect to X_i by $\text{deg}(f, i)$, i.e., for a term α , $\text{deg}(\alpha, i) = a_i$ if $X_i^{a_i}$ divides α and if $X_i^{a_i+1}$ does not divide α . We also denote the multi-degree of a term $\alpha \in T(\bar{X})$ by $\text{deg}(\alpha) = (a_1, \dots, a_n)$ when $\alpha = X_1^{a_1} \cdots X_n^{a_n} \in T(\bar{X})$.

For non-negative real numbers a_1, \dots, a_n and $\alpha \in T(\bar{X})$, we denote $\text{tdeg}_{(a_1, \dots, a_n)}(\alpha) \in \mathbb{R}_{\geq 0}$ by $\text{tdeg}_{(a_1, \dots, a_n)}(\alpha) = a_1 \cdot \text{deg}(\alpha, 1) + \dots + a_n \cdot \text{deg}(\alpha, n)$. For $f \in K[\bar{X}]$, we define $\text{tdeg}_{\bar{a}}(f) = \max\{\text{tdeg}_{\bar{a}}(\alpha) : \alpha \in \text{terms}(f)\}$.

For $F \subseteq K[\bar{X}]$, we denote $\langle F \rangle_K \subseteq K[\bar{X}]$ by $\langle F \rangle_K = \{a_1 f_1 + \dots + a_m f_m : a_1, \dots, a_m \in K, f_1, \dots, f_m \in F\}$ and $\langle F \rangle_{K[\bar{X}]} \subseteq K[\bar{X}]$ by $\langle F \rangle_{K[\bar{X}]} = \{g_1 f_1 + \dots + g_m f_m : g_1, \dots, g_m \in K[\bar{X}], f_1, \dots, f_m \in F\}$.

Our plan is as below. In Section 2, we give the main Lemma which gives a sufficient condition to get a Gröbner basis. In Section 3 and 4, we give several definitions and properties used in the main algorithm. In Section 5, we argue on the dynamic change of term order. In Section 6, we give the main algorithm to compute Gröbner bases along with the corresponding term order. In Section 7, we show several optimization techniques. In Section 8, we show a prototypical implementation with a few tiny computational examples.

2 A Sufficient Condition to Get a Gröbner Basis

For an n -tuple $\bar{b} = (b_1, \dots, b_n)$ of non-negative integers, we define $\text{TB}(\bar{b}) = \{\alpha \in T(\bar{X}) : \text{deg}(\alpha, i) \leq b_i \text{ for each } i = 1, \dots, n\}$. In this section, we identify polynomials in $K[\bar{X}]$ whose terms are in $\text{TB}(\bar{b})$ with linear combinations of $\text{TB}(\bar{b})$, i.e., $\langle \text{TB}(\bar{b}) \rangle_K = \{f \in K[\bar{X}] : \text{terms}(f) \subseteq \text{TB}(\bar{b})\}$, and express being Gröbner bases in linear space $\langle \text{TB}(\bar{b}) \rangle_K$.

Definition 1. For $F \subseteq K[\bar{X}]$ and a term order \prec on $T(\bar{X})$, we say F is K -reduced with respect to \prec if $\text{ht}_\prec(g) \notin \text{terms}(f)$ for any distinct $f, g \in F$.

Lemma 2. Fix a term order \prec on $T(\bar{X})$. Assume $G \subseteq K[\bar{X}]$ is K -reduced and $f \in \langle G \rangle_K \setminus \{0\}$. Then f is top- K -reducible modulo G wrt \prec , i.e., there is $g \in G$ and $a \in K \setminus \{0\}$ such that $\text{ht}_\prec(f + ag) < \text{ht}_\prec(f)$.

Proof. We say $\alpha = \text{ht}(f)$. Since $f \in \langle G \rangle_K$, we can take distinct $g_1, \dots, g_m \in G$ and $a_1, \dots, a_m \in K \setminus \{0\}$ such that $f = a_1 g_1 + \dots + a_m g_m$. We assume, for $i = 1, \dots, m$, that $\text{ht}(g_i) \neq \alpha$ for a contradiction.

Then we can take g_k such that $\alpha \in \text{terms}(g_k)$ since $\alpha \in \text{terms}(f)$. So letting $\beta = \text{ht}(g_k)$, we have $\alpha < \beta$ since $\text{ht}(g_k) \neq \alpha$. Then there is g_i such that $g_i \neq g_k$ and that $\beta \in \text{terms}(g_i)$ since $\beta \notin \text{terms}(f)$, which contradicts to the assumption that G is K -reduced.

So there is g_k such that $\text{ht}(g_k) = \alpha = \text{ht}(f)$. Let $a = -\text{hc}(f) \cdot \text{hc}(g_k)^{-1} \in K \setminus \{0\}$. Then we have $g_k \in G$ and $\text{ht}(f + ag_k) < \text{ht}(f)$. \square

The following is just a generalization of usual Gröbner bases on polynomial rings. We notice that G is a Gröbner basis in the usual meaning when F is an ideal in $K[\bar{X}]$.

Definition 3. For $F, G \subseteq K[\bar{X}]$ and a term order $<$, we say G is a Gröbner basis of F with respect to $<$ if $\langle G \rangle_{K[\bar{X}]} \subseteq \langle F \rangle_{K[\bar{X}]}$ and if, for every $f \in F \setminus \{0\}$, there is $g \in G$ such that $\text{ht}_{<}(g)$ divides $\text{ht}_{<}(f)$. We say G is a Gröbner basis when G is a Gröbner basis of $\langle G \rangle_{K[\bar{X}]}$.

Definition 4. (*T-closed*) For $T \subseteq T(\bar{X})$ and $F \subseteq \langle T \rangle_K$, we say F is *T-closed* if $\alpha f \in \langle F \rangle_K$ for any $\alpha \in T(\bar{X})$ and $f \in F$ such that $\text{terms}(\alpha f) \subseteq T$.

Notation 5. (*side*) Let $<$ be a term order on $T(\bar{X})$. We define a mapping $\text{side}_{<} : K[\bar{X}] \times \{1, \dots, n\} \rightarrow \mathbb{Z}_{\geq 0}$ by $\text{side}_{<}(f, i) = \max\{\text{deg}(\alpha, i) - \text{deg}(\text{ht}_{<}(f), i) : \alpha \in \text{terms}(f)\}$.

Now we have the following Lemma. In the latter sections, we give several algorithms, and show ways to get Gröbner basis using this Lemma.

Lemma 6. Fix some term order $<$ on $T(\bar{X})$. Let $\bar{b} = (b_1, \dots, b_n)$ and $\bar{c} = (c_1, \dots, c_n)$ are n -tuples of non-negative integers. Let $G \subseteq \langle \text{TB}(\bar{b}) \rangle_K$ be $\text{TB}(\bar{b})$ -closed and K -reduced wrt $<$. Let $G_0 \subseteq \langle \text{TB}(\bar{c}) \rangle_K$ be a Gröbner basis of G such that $G_0 \subseteq G$. Assume, for any $g \in G_0$ and $i = 1, \dots, n$, that $c_i + \text{side}_{<}(g, i) \leq b_i$. Then G_0 is a Gröbner basis wrt $<$.

Proof. We pick $f, g \in G_0$ with $f \neq g$ arbitrarily. We show that $\text{SPol}(f, g) \xrightarrow{*}_{G_0} 0$. Let $\alpha = \text{ht}(f)$, $a = \text{hc}(f)$, $\beta = \text{ht}(g)$, $b = \text{hc}(g)$, $\gamma = \text{lcm}(\alpha, \beta)$. Then $\text{SPol}(f, g) = b(\gamma/\alpha)f - a(\gamma/\beta)g$.

We first show that $b(\gamma/\alpha)f, a(\gamma/\beta)g \in \langle G \rangle_K$. We say $\text{deg}(\alpha) = (a_1, \dots, a_n)$ and $\text{deg}(\beta) = (a'_1, \dots, a'_n)$. Pick arbitrary $\delta \in \text{terms}(f)$ and say $\text{deg}(\delta) = (d_1, \dots, d_n)$. Then $\text{deg}((\gamma/\alpha)\delta) = (\max(a_1, a'_1) - a_1 + d_1, \dots, \max(a_n, a'_n) - a_n + d_n)$. For each $i = 1, \dots, n$, since $d_i - a_i \leq \text{side}(f, i)$ and $\max(a_i, a'_i) \leq c_i$, we have $\max(a_i, a'_i) - a_i + d_i \leq c_i + \text{side}(f, i) \leq b_i$. Thus $(\gamma/\alpha)\delta \in \text{TB}(\bar{b})$. Since G is $\text{TB}(\bar{b})$ -closed, we have $b(\gamma/\alpha)f \in \langle G \rangle_K$. By the same way, we can show that $a(\gamma/\beta)g \in \langle G \rangle_K$.

So letting $s_0 = \text{SPol}(f, g)$, we have $s_0 \in \langle G \rangle_K$. Next we construct sequences $s_0, \dots, s_l, s_{l+1} \in \langle G \rangle_K$ and $g_0, \dots, g_l \in G_0$ such that $\text{ht}(s_i) > \text{ht}(s_{i+1})$ for each i . Assume we have $s_i \in \langle G \rangle_K$. Then, since s_i is top- K -reducible modulo G , we have $g' \in G$ and $c \in K \setminus \{0\}$ such that $\text{ht}(s_i + cg') < \text{ht}(s_i)$. Since G_0 is a Gröbner basis of G , we can take $g_i \in G_0$ and $\eta \in T(\bar{X})$ such that $\text{ht}(\eta g_i) = \text{ht}(g')$. Let $d \in K$ be such that $\text{hm}(cg') = \text{hm}(d\eta g_i)$. We set $s_{i+1} = s_i + d\eta g_i$. Then we see that $\text{ht}(s_{i+1}) < \text{ht}(s_i)$. From the fact $d\eta g_i \in \langle G \rangle_K$, we also see that $s_{i+1} \in \langle G \rangle_K$.

Since $<$ is Noetherian, we have $s_{l+1} = 0$ for some l . Thus we arrive at $\text{SPol}(f, g) \xrightarrow{*}_{G_0} 0$. \square

In the rest of this paper, we give a procedure to compute G required at the assumption of this Lemma.

3 T-Closed K-Reduced Bases

In this section, we show a way to compute $TB(\bar{d})$ -closed and K -reduced basis for a given ideal on $K[\bar{X}]$ and an n -tuple \bar{d} of non-negative integers.

Notation 7. For $T \subseteq T(\bar{X})$ and $F \subseteq K[\bar{X}]$, we let $\text{mult}(F, T) = \{\alpha \cdot f \in K[\bar{X}] : \alpha \in T(\bar{X}), f \in F, \text{terms}(\alpha f) \subseteq T\}$.

We should notice that, if both of $F \subseteq K[\bar{X}]$ and T are finite, $\text{mult}(F, T)$ is finite. From Lemma 2, we also notice that, if $\langle F \rangle_K = \langle G \rangle_K$ and G is K -reduced, then $\text{ht}[F] \subseteq \text{ht}[G]$. We may assume that an algorithm `LinearReduce` is given, where, for $F \subseteq K[\bar{X}]$ and a well-order $<$, if $B := \text{LinearReduce}(F, <)$, then B is K -reduced with respect to $<$, $\langle B \rangle_K = \langle F \rangle_K$. We should note that it is essentially same as Gaussian elimination. Then we have the following algorithm.

Algorithm. CloseAndLReduce

Input: F : a finite subset of $K[\bar{X}]$,
 T : a finite subset of $T(\bar{X})$ with $F \subseteq \langle T \rangle_K$,
 $<$: a term order on $T(\bar{X})$
Output: R : a T -closed and K -reduced subset of $\langle T \rangle_K$
wrt $<$ such that $\langle F \rangle_{K[\bar{X}]} = \langle R \rangle_{K[\bar{X}]}$

```

R := F;
d := |R|;
d' := 0;
while d' ≠ d do
    d' := d;
    R := LinearReduce(mult(R, T), <);
    d := |R|;
end while
return R;
end.
```

Lemma 8. Let T be a finite subset of $T(\bar{X})$, F be a finite subset of $\langle T \rangle_K$, and $<$ be a term order on $T(\bar{X})$. Then the algorithm `CloseAndLReduce` terminates in finite steps and, the output R of `CloseAndLReduce`($F, T, <$) satisfies that R is T -closed and K -reduced wrt $<$ and that $\langle F \rangle_{K[\bar{X}]} = \langle R \rangle_{K[\bar{X}]}$.

Proof. Let R_n be the R in the n -th while-iteration in the algorithm. Then we see that $\langle R_n \rangle_K \subsetneq \langle R_{n+1} \rangle_K$ and so $\dim_K \langle R_n \rangle_K < \dim_K \langle R_{n+1} \rangle_K$ for each n . Since $\dim_K \langle R_n \rangle_K \leq |T|$, the algorithm terminates in a finite step.

Let R be the output and pick $f \in R$ arbitrarily. Let $\alpha \in T(\bar{X})$ be such that $\text{terms}(\alpha f) \subseteq T$. Then $\alpha f \in \text{mult}(R, T)$ by the definition of $\text{mult}(R, T)$. Let R' be the output of `LinearReduce`($\text{mult}(R, T), <$). Then $\langle R \rangle_K = \langle R' \rangle_K$ by the definition of algorithm. Since $\alpha f \in \langle R' \rangle_K = \langle R \rangle_K$, we see that R is T -closed.

We can easily see that R is K -reduced wrt $<$. □

4 Finding a Candidate

We fix an algorithm `Bound` such that, for finite $F \subseteq K[\bar{X}]$, $\bar{b} = \text{Bound}(F)$ if $b_i = \max\{\deg(f, i) : f \in F\}$ for $i = 1, \dots, n$ where $\bar{b} = (b_1, \dots, b_n)$. Then $F \subseteq \langle \text{TB}(\text{Bound}(F)) \rangle_K$. Then, using the algorithm `CloseAndLReduce`, we can get $G \subseteq K[\bar{X}]$ such that $\langle F \rangle_{K[\bar{X}]} = \langle G \rangle_{K[\bar{X}]}$ and that G is $\text{TB}(\bar{b})$ -closed and K -reduced. If we can find $G_0 \subseteq G$ which satisfies the assumptions of Lemma 6, it is a Gröbner basis of $\langle F \rangle_{K[\bar{X}]}$. So, we give a procedure to calculate a candidate of G_0 as below.

Algorithm. MinimalBasis

Input: $G \subseteq K[\bar{X}]$ finite, $<$: term order on $T(\bar{X})$

Output: $H \subseteq G$ a Gröbner basis of G

```

H := ∅;
while G ≠ ∅ do
  g := min<G;
  G := G \ {g};
  is_gb := true;
  foreach h ∈ H do
    if ht<(h)|ht<(g) then
      is_gb := false;
    end if
  end for
  if is_gb then
    H := H ∪ {g};
  end if
end for
return H;
end.

```

For some $\bar{b} \in (\mathbb{Z}_{\geq 0})^n$ and $G \subseteq \text{TB}(\bar{b})$ which is $\text{TB}(\bar{b})$ -closed and K -reduces wrt $<$, let $G_0 := \text{MinimalBasis}(G, <)$ and $\bar{c} := \text{Bound}(G_0)$. If they satisfy the assumption of Lemma 6, we arrive at the destination, i.e., G_0 is a required Gröbner basis of the given ideal. But if they do not satisfies it, we have to extend the bound \bar{b} . We should note that, when we extend the bound \bar{b} to \bar{c} for $G \subseteq \langle \text{TB}(\bar{b}) \rangle$, we have to compute $\text{mult}(G, \text{TB}(\bar{c}))$.

If the bound \bar{b} changes to another value, the term order for an efficient computation of $\text{TB}(\bar{b})$ -closed K -reduced basis would also be changed. In the next section, we give a guideline to set an appropriate term order.

5 Appropriate Term Order

Combining the procedures above, we can compute a Gröbner basis for a given ideal with respect to some given term order by extending the bound \bar{b} in each

step. On the other hand, we can switch term order during computation dynamically in order to make the dimension $|\text{TB}(\bar{b})|$ of the linear space at the next step small. For such a purpose, we introduce weighted term order.

Definition 9. Let $<$ be a term order on $T(\bar{X})$ and \bar{w} be an n -tuple of non-negative reals. Then we define the order $<_{\bar{w}}$ weighted by \bar{w} coherent to $<$ by, for any $\alpha, \beta \in T(\bar{X})$,

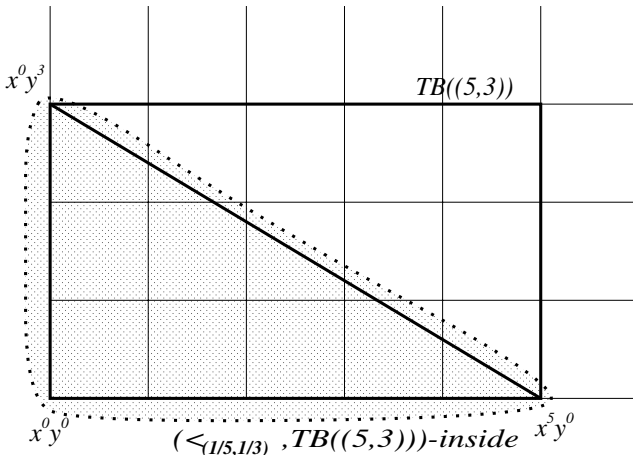
1. if $\text{tdeg}_{\bar{w}}(\alpha) < \text{tdeg}_{\bar{w}}(\beta)$, then $\alpha <_{\bar{w}} \beta$, and
2. if $\text{tdeg}_{\bar{w}}(\alpha) = \text{tdeg}_{\bar{w}}(\beta)$ and if $\alpha < \beta$, then $\alpha <_{\bar{w}} \beta$.

For any term order $<$ on $T(\bar{X})$ and weights $\bar{w} \in (\mathbb{R}_{\geq 0})^n$, we easily see that $<_{\bar{w}}$ forms a term order on $T(\bar{X})$.

For $b_1, \dots, b_n \in \mathbb{Z}_{\geq 0}$, we let $T = \text{TB}((b_1, \dots, b_n))$. For a term order $<$, we say a term $\alpha \in T$ is $(<, T)$ -inside if $\beta \in T$ for all $\beta < \alpha$. Then we notice that, if the head term $\text{ht}_{<}(f)$ of a polynomial is $(<, T)$ -inside, reduction of f modulo some g can be computed within T . So choosing a term order $<$ to make the number of $(<, T)$ -inside terms large, we can make the computation of Gröbner basis within T efficiently.

For such a purpose, we are going to choose an appropriate degree $d \in \mathbb{R}_{> 0}$ and weights $\bar{w} = (w_1, \dots, w_n)$ for some term order $<$ for making the number large of the terms $\alpha \in T(\bar{X})$ with $\text{tdeg}_{\bar{w}}(\alpha) \leq d \rightarrow \alpha \in T$. Even if some b_i equals to 0, the corresponding weight w_i does not affect to $(<_{\bar{w}}, T)$ -insideness, and so we may assume that every b_i 's are positive. Since $X_1^{b_1} X_2^0 \cdots X_n^0 \in T = \text{TB}((b_1, \dots, b_n))$, we have $d \leq w_1 b_1$. By the same way, since $X_1^0 X_1^0 \cdots X_i^{b_i} \cdots X_{n-1}^0 X_n^0 \in T$, we have $d \leq w_i b_i$ for each $i = 1, \dots, n$. Thus we have $d \leq \min\{w_1 b_1, \dots, w_n b_n\}$. Then we notice that we can get the maximum number of the terms α with $\text{tdeg}_{\bar{w}}(\alpha) \leq d \rightarrow \alpha \in T$ when $d = w_1 b_1 = \dots = w_n b_n$.

So we define the algorithm `WeightedOrder` to give appropriate term order to compute within given $\text{TB}(\bar{d})$ by $< = \text{WeightedOrder}((b_1, \dots, b_n), <)$ if $< = <_{(w_1, \dots, w_n)}$ where $w_i = 1/b_i$ if $b_i \neq 0$ and $w_i = 0$ if $b_i = 0$ for $i = 1, \dots, n$.



6 The Main Algorithm

Now we can describe an algorithm to compute Gröbner basis as below. The essential part of the computation is in **CloseAndLReduce**, and it is a combination of **mult()** and **LinearReduce**. In fact, the Buchberger algorithm consists of S-polynomials and monomial reductions, and they consist of multiplications of coefficients and terms to polynomial and additions of two polynomials. We may realize that the part of the multiplications of terms to polynomials is included in **mult()**, and the one of the multiplications of coefficients to polynomials and additions of two polynomials is in **LinearReduce**, Gaussian elimination.

Algorithm. GroebnerBasisLA

Input: $F \subseteq K[\bar{X}]$: finite

Output: $G \subseteq K[\bar{X}]$: a Gröbner basis of $\langle F \rangle_{K[\bar{X}]}$ wrt \prec ,

\prec : a term order on $T(\bar{X})$ let $<$ be a grlex order;

$(b_1, \dots, b_n) := (0, \dots, 0)$;

$(c_1, \dots, c_n) := \text{Bound}(F)$;

$(d_1, \dots, d_n) := (0, \dots, 0)$;

$G := \text{mult}(F, \text{TB}((c_1, \dots, c_n)))$;

$T := \text{terms}(F)$;

$\prec := \text{WeightedOrder}((c_1, \dots, c_n), <)$;

while $c_1 + d_1 > b_1$ **or** ... **or** $c_n + d_n > b_n$ **do**

for $i = 1, \dots, n$ **do**

$b_i := c_i + d_i$;

end for

$G := \text{CloseAndLReduce}(G, \text{TB}((b_1, \dots, b_n)), \prec)$;

$G_0 := \text{MinimalBasis}(G, \prec)$;

$\prec := \text{WeightedOrder}(\text{Bound}(G_0), <)$;

$\alpha_T := \max_{\prec}\{T\}$;

$G_1 := \{g \in G : \text{terms}(g) \preceq \alpha_T\}$;

$(c_1, \dots, c_n) := \text{Bound}(G_0 \cup G_1)$;

for $i = 1, \dots, n$ **do**

$d_i := \max\{\text{side}_{\prec}(g, i) : g \in G_0\}$;

end for

end while

return (G_0, \prec) ;

end.

Then we have the following Theorem.

Theorem 10. *We assume that F is a finite subset of $K[\bar{X}]$. Then the algorithm $\text{GroebnerBasisLA}(F)$ terminates in finite steps. So we also let (G_0, \prec) be the*

output of `GroebnerBasisLA(F)`. Then G_0 is a Gröbner basis of $\langle F \rangle_{K[\bar{X}]}$ with respect to \prec .

Proof. Let G' be a universal Gröbner basis of $\langle F \rangle_{K[\bar{X}]}$. For each $i = 1, \dots, n$, we let $c'_i = \max\{\deg(g, i) : g \in G'\}$, $d'_i = \max\{\deg(\alpha, i) - \deg(\beta, i) : \alpha, \beta \text{ terms}(g) \text{ for some } g \in G'\}$, and $b_i = c'_i + d'_i$. We let \prec' be the output of `WeightedOrder` $((b_1, \dots, b_n), \prec)$ where \prec is the grlex order used in the algorithm. Then we let G be the output of `CloseAndLReduce` $(G, \text{TB}(b_1, \dots, b_n), \prec')$ and $G_0, (c_1, \dots, c_n)$, and (d_1, \dots, d_n) be as in the algorithm. Then we see that $c_i + d_i \leq b_i$ for every $i = 1, \dots, n$. So the while-loop terminates at most $(b_1 + 1) \cdot (b_2 + 1) \cdots (b_n + 1)$ -many steps.

Next we see, at the end of the algorithm, that $G_0 \cup G_1$ forms a Gröbner basis of $\langle F \rangle_{K[\bar{X}]}$ with respect to \prec by Lemma 6. In fact, letting \prec, G, G_0, G_1 and $b_1, c_1, d_1, \dots, b_n, c_n, d_n$ be as in the end of the algorithm, we see, for any $i = 1, \dots, n$ and $g \in G_0 \cup G_1$, that $\text{side}_\prec(g, i) \leq d_i$ and so $c_i + \text{side}_\prec(g, i) \leq b_i$. Thus they satisfy the assumption of the Lemma and so $G_0 \cup G_1$ is a Gröbner basis. Next we let α_T be as in the end of algorithm. Since $F \subseteq \langle \alpha \in T(\bar{X}) : \alpha \preceq \alpha_T \rangle_K$, we see that $F \subseteq \langle G_1 \rangle_K$, and so $F \subseteq \langle G_0 \cup G_1 \rangle_{K[\bar{X}]}$. Thus $G_0 \cup G_1$ is a Gröbner basis of $\langle F \rangle_{K[\bar{X}]}$ with respect to \prec .

By the definition of `MinimalBasis`, we see that G_0 equals to `MinimalBasis` $(G_0 \cup G_1, \prec)$. Thus we see that G_0 is a Gröbner basis of $\langle F \rangle_{K[\bar{X}]}$ with respect to \prec . □

Note. In fact, the algorithm `GroebnerBasisLA` outputs the reduced Gröbner basis if `LinearReduce` outputs reduced row echelon forms, though we omit the proof.

7 Optimizations

This algorithm is heavily depend on `mult()` and `LinearReduce`. In this section, we introduce two methods to optimize them. Their basic concepts are in (1) eliminating duplicated polynomials in the output of `mult()`, and (2) reducing surplus polynomials in the input of the algorithm `LinearReduce`.

7.1 Extending Bound

We consider the case that $F \subseteq \langle \text{TB}((b_1, \dots, b_n)) \rangle_K$ is $\text{TB}((c_1, \dots, c_n))$ -closed for some $c_i \leq b_i$ ($i = 1, \dots, n$). When we calculate $\text{TB}((b_1, \dots, b_n))$ -closure of F , we can get it by `mult(F, TB((b_1, \dots, b_n)))` but it contains lots of duplicated polynomials. Then, using the information that F is $\text{TB}((c_1, \dots, c_n))$ -closed, we notice that it is enough to calculate `mult(f, TB((b_1, \dots, b_n)))` only for $f \in F$ on the “bound” of $\text{TB}((c_1, \dots, c_n))$. Thus we introduce the following algorithm `ExtendBound`, though we omit the proof of its validity.

Algorithm. ExtendBound

Input: $F \subseteq K[\bar{X}]$, $(b_1, \dots, b_n), (c_1, \dots, c_n) \in \mathbb{Z}_{\leq 0}^n$ such that F is $\text{TB}((c_1, \dots, c_n))$ -closed and $c_i \leq b_i$ for $i = 1, \dots, n$.
Output: $G \subseteq K[\bar{X}]$ such that G is $\text{TB}((b_1, \dots, b_n))$ -closed and $\langle G \rangle_{K[\bar{X}]} = \langle F \rangle_{K[\bar{X}]}$.

```

G := F;
foreach i = 1, ..., n do
  B := {f ∈ G : deg(f, i) = c_i};
  foreach f ∈ B and j = 1, ..., b_n - c_n do
    G := G ∪ {X_i^j f};
  end for
end for
return G;
end.

```

7.2 Reduction Matrix

In this paper, we identify a K -linear combination of $\text{TB}((b_1, \dots, b_n))$ with a polynomial in $K[\bar{X}]$ whose terms are in $\text{TB}((b_1, \dots, b_n))$. We fix $\bar{b} = (b_1, \dots, b_n)$ and a term-order \prec on $T(\bar{X})$, let $1 = t_1 \prec t_2 \prec \dots \prec t_l$ be the enumeration of $\text{TB}(\bar{b})$ where $l = (b_1 + 1) \cdot (b_2 + 1) \cdot \dots \cdot (b_n + 1)$. Let $\phi: \langle t_1, \dots, t_l \rangle_K \rightarrow K^l$ be the canonical isomorphism.

For $g \in \langle \text{TB}(\bar{b}) \rangle$, let k_g and c_g be such that $t_{k_g} = \text{ht}_{\prec}(g)$ and $c_g = \text{hc}_{\prec}(g) \in K \setminus \{0\}$. Then we define the *reduction matrix* $R_g = (r_{ij})_{1 \leq i, j \leq l}$ induced by g by $r_{ik_g} = -\phi(g)_i/c_g$ if $i > k_g$, $r_{ii} = 1$ if $i \neq k_g$, and $r_{ij} = 0$ otherwise, where $\phi(g)_i \in K$ is the i -th component of the vector $\phi(g)$, i.e., the coefficient of t_i occurring in g . Then we notice, for any $f \in \langle \text{TB}(\bar{b}) \rangle_K$, letting $f' = \phi^{-1}(R_g \cdot \phi(f))$, that $f \xrightarrow{*}_G f'$. In fact, saying $f = a_1 t_1 + \dots + a_l t_l$, we can easily check that $\phi^{-1}(R_g \cdot \phi(a_j t_j)) = a_j t_j$ for $j \neq k_g$ and $\phi^{-1}(R_g \cdot \phi(a_{k_g} t_{k_g})) = -a_{k_g} \cdot (\phi(g - \text{hm}_{\prec}(g))/\text{hc}_{\prec}(g))$, and so $\phi^{-1}(R_g \cdot \phi(f)) = f - a_{k_g} t_{k_g} - a_{k_g} \cdot \text{hc}_{\prec}(g)^{-1} \cdot (g - \text{hm}_{\prec}(g)) = f - a_{k_g} \cdot \text{hc}_{\prec}(g)^{-1} \cdot g$.

Furthermore, we let G be a K -reduced subset of $\langle \text{TB}(\bar{b}) \rangle_K$. Then we can also define the *reduction matrix* R_G induced by G by the same fashion as in the previous paragraph. For each $g \in G$, we let $1 \leq k_g \leq l$ and $c_g \in K \setminus \{0\}$ be as above. Then we define $R_G = (r_{ij})_{1 \leq i, j \leq l}$ by

$$r_{ij} = \begin{cases} -\phi(g)_i/c_g, & \text{if } i > \text{ and } j = k_g \text{ for some } g \in G, \\ 1, & \text{if } i = j \text{ and } j \neq k_g \text{ for any } g \in G, \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

We can check that R_G is well-defined and, for any $f \in \langle \text{TB}(\bar{b}) \rangle_K$, that $f' = \phi^{-1}(R_G \cdot \phi(f))$ satisfies $f \xrightarrow{*}_G f'$ and f' is not K -reducible modulo G . Especially, for a K -reduced finite subset G of $\langle \text{TB}(\bar{b}) \rangle_K$ and a polynomial $f \in \langle \text{TB}(\bar{b}) \rangle_K$, we can check whether $f \in \langle G \rangle_K$ or not using the reduction matrix R_G induced by G .

7.3 Optimized CloseAndLReduce

In the algorithm `GroebnerBasisLA`, the input G of `CloseAndLReduce` is always $\text{TB}(\bar{c})$ -closed for some \bar{c} . So, placing the call to the algorithm `ExtendBound` at the beginning of the algorithm `CloseAndLReduce`, we can assume that the candidate $R \subseteq \langle \text{TB}((b_1, \dots, b_n)) \rangle_K$ in an execution of `CloseAndLReduce`($G, \text{TB}((b_1, \dots, b_n)), <$) is $\text{TB}((b_1, \dots, b_n))$ -closed in the rest of the computing, thus we can optimize it as follow.

Algorithm. `CloseAndLReduce` (optimized)

Input: F : a finite subset of $K[\bar{X}]$,

\bar{b} : an n -tuple of non-negative integers with $F \subseteq \langle \text{TB}(\bar{b}) \rangle_K$,

$<$: a term order on $T(\bar{X})$

Output: G : a T -closed and K -reduced subset of $\langle T \rangle_K$

wrt $<$ such that $\langle F \rangle_{K[\bar{X}]} = \langle G \rangle_{K[\bar{X}]}$

$\bar{c} := \text{Bound}(F)$;

$H := \text{ExtendBound}(F, \bar{c}, \bar{b})$;

$H' := \emptyset$;

$G := \emptyset$;

$d := |F|$;

$d' := 0$;

while $d' \neq d$ do

$d' := d$;

$G := H \setminus G$;

 let $R_{H'}$ be the reduction matrix induced by H' ;

$G' := \{R_{H'} \cdot g : g \in \text{mult}(G, \text{TB}(\bar{b}))\}$;

$H := \text{LinearReduce}(G', <)$;

$H' := H$;

$d := |H|$;

end while

return G ;

end.

8 An Implementation

We choose PARI/GP¹ to implement the algorithms in this paper in order to demonstrate that it is not difficult to implement Gröbner bases computation to a system which has neither S-polynomials nor monomial reductions. You can download the file from our page². This file has three main routines, (1) `groebner_basis_la`, (2) `groebner_basis_la_opt`, and (3) `groebner_basis_la_mat`. The first one corresponds to the Algorithm `GroebnerBasisLA` in Section 6 and the second one is an implementation applying the optimization methods

¹ <http://pari.math.u-bordeaux.fr/>

² <http://kurt.scitec.kobe-u.ac.jp/~sakira/GBwithinLA/>

in Section 7. The last (3) is a one which we remove the use of `LinearReduce`. Though we omit the details, the technique of reduction matrix in Subsection 7.2 can be a substitution for `LinearReduce` in order to get a Gröbner basis by the idea described in this paper. In these main algorithms input polynomials are converted to vectors in a direct product of \mathbb{Q} just before of computation, the computed vectors are converted to output polynomials at the end of the routines, and most of all computations are processed within linear algebra.

We give a tiny computational example as follow: For an example, when we put `groebner_basis_la_opt([x^2+y^2+z^2-20, x+y-5, x*y*z-3])`, it outputs `[[8*z^3+40*z-48, 2*y^2-10*y+(z^2+5), x+(y-5)], [x,y,z], [6, 3, 2]]` which means $\{z^3+5z-6, 2y^2-10y+z^2+5, x+y-5\}$ is the reduced Gröbner basis of $\langle x^2+y^2+z^2-20, x+y-5, xyz-3 \rangle_{\mathbb{Q}[x,y,z]}$ with respect to the term-order weighted by $\{x : 6, y : 3, z : 2\}$. In the following table, we give a timing data of these implementations though they are not faster than existing implementations to compute Gröbner bases. The unit of time in the table is the second. (Mac OS X 10.5.6, CPU 2.8GHz Xeon, Memory 22GB, GP/PARI 2.3.4 with x86-64/GMP-4.2.3)

polynomial system	(1) no opt.	(2) opt.	(3) red. mat.
$\{xy + z - xy, x^2 - z, x^3 - x^2yz\}$	10.5	1.5	10.6
$\{xy + z - xz, x^2 - z, 2x^3 - x^2yz - 1\}$	> 1 hour	1.6	11.5
$\{5x^3 - 7yz, 11y^2 - 101z, x + y - 65537z\}$	> 1 hour	9.3	0.8
$\{x^2 + y^2 + z^2 - 20, x + y - 5, xyz - 3\}$	> 1 hour	3.7	12.1

The routine `groebner_basis_la_opt` is the fastest in these ones, though all of them can calculate very tiny polynomial systems at the current moment.

9 Conclusion and Remarks

The algorithm `GroebnerBasisLA` does not require a term order as its input. It is one of the characteristic properties of the algorithm, and it dynamically chooses a suitable term order for the set of terms appearing in the linear space for the sake of its efficient computation. For a given finite set F of polynomials in $K[\bar{X}]$, our algorithm compute a Gröbner basis G of $\langle F \rangle_{K[\bar{X}]}$ with the corresponding term order on $T(\bar{X})$. On the other hand, if we need the Gröbner basis with respect to a given term order, we can use a method for change of order, e.g., Gröbner walk [5], FGLM [8], and Hilbert driven [16].

Another of the characteristic properties is on lacking both of S-polynomials and monomial reductions. Though it does not have them explicitly, the combination of `mult()` and `LinearReduce` involves them. So the algorithm places more weight on Gaussian elimination than the other algorithms to compute Gröbner bases. In our method, we require a set G of polynomials in $\langle \text{TB}(\bar{b}) \rangle_K$ to be $\text{TB}(\bar{b})$ -closed, and it may spend many memory resources. From another viewpoint, we may consider that much intermediary information is stored in memory, and so

the computation speed may become faster for some case if we have enough memory resources. We expect this tendency would be emphasized if the coefficient ring K is finite, since less additional allocation of memory is required during the computation in a fixed bound $TB(\bar{b})$. We also expect to be used several improved algorithms for linear algebra, e.g., parallel Gaussian elimination, in our algorithm. Since PARI/GP does not support neither parallelization methods nor improved algorithms for Gaussian eliminations, we shall implement our algorithm to another computer algebra system with such features.

References

1. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. *J. Symb. Comput.* 24(3-4), 235–265 (1997)
2. Brickenstein, M.: Slimgb: Gröbner bases with slim polynomials. Reports on Computer Algebra 35, ZCA, University of Kaiserslautern (2005)
3. Buchberger, B.: Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems. *Aequ. Math.* 4(3), 374–383 (1970)
4. Caboara, M.: A dynamic algorithm for Gröbner basis computation. In: Proc. ISSAC 1993, pp. 275–283 (1993)
5. Collart, S., Kalkbrenner, M., Mall, D.: Converting Bases with the Gröbner Walk. *J. Symb. Comput.* 24(3-4), 465–469 (1997)
6. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F_4). *J. Pure and Applied Algebra* 139(1–3), 61–88 (1999)
7. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In: Proc. ISSAC 2002, pp. 75–83 (2002)
8. Faugère, J.C., Gianni, P.M., Lazard, D., Mora, T.: Efficient Computation of Zero Dimensional Gröbner Bases by Change of Ordering. *J. Symb. Comput.* 16(4), 329–344 (1993)
9. Gebauer, R., Möller, H.M.: On an installation of Buchberger’s algorithm. *J. Symb. Comput.* 6(2-3), 275–286 (1988)
10. Giovini, A., Mora, T., Nielsi, G., Robbiano, L., Traverso, C.: One sugar cube, please! OR Selection strategies in the Buchberger algorithm. In: Proc. ISSAC 1991, pp. 49–54 (1991)
11. Greuel, G.M., Pfister, G.: SINGULAR and Applications. Jahresbericht der DMV 108, 167–196 (2006)
12. Lazard, D.: Gröbner-Bases, Gaussian elimination and resolution of systems of algebraic equations. In: van Hulzen, J.A. (ed.) ISSAC 1983 and EUROCAL 1983. LNCS, vol. 162, pp. 146–156. Springer, Heidelberg (1983)
13. Mora, T., Robbiano, L.: The Gröbner fan of an ideal. *J. Symb. Comput.* 6(2-3), 183–208 (1988)
14. Noro, M., Takeshima, T.: Risa/Asir – A Computer Algebra System. In: Proc. ISSAC 1992, pp. 387–396 (1992)
15. Traverso, C.: Gröbner trace algorithms. In: Gianni, P. (ed.) ISSAC 1988. LNCS, vol. 358, pp. 125–138. Springer, Heidelberg (1989)
16. Traverso, C.: Hilbert functions and the Buchberger algorithm. *J. Symb. Comput.* 22(4), 355–376 (1996)

A Mimetic Finite-Difference Scheme for Convection of Multicomponent Fluid in a Porous Medium

Vyacheslav Tsybulin¹, Andrew Nemtsev², and Bülent Karasözen³

¹ Southern Federal University, Rostov-on-Don, Russia

² Southern Scientific Center of Russian Academy of Science, Rostov-on-Don, Russia

³ Department of Mathematics & Institute of Applied Mathematics, Middle East Technical University, Ankara, Turkey

Abstract. A mimetic finite-difference scheme for the equations of three-dimensional convection of a multicomponent fluid in a porous medium is developed. The discretization is based on staggered grids with five types of nodes (velocities, pressure, temperature, and mass fractions) and on a special approximation of nonlinear terms. Computer experiments have revealed the continuous family of steady states in the case of the zero heat fluxes through two opposite lateral planes of parallelepiped.

Introduction

Natural convection of an incompressible fluid in a porous medium differs from that of single phase fluid [1]. Usually, after the state of rest, a finite number of regimes (convective patterns) may appear due to loss of stability. An exciting example with an infinite number of steady states was found for the planar problem of incompressible fluid convection in a porous medium [2]. This phenomenon of appearance of a continuum of solutions was explained by the cosymmetry theory [3]. The strong nonuniqueness of steady states in the planar Darcy convection was proved for the single [3] and multicomponent fluid [10]. First computations of the families in the planar Darcy convection were done by the Galerkin method [4] and the finite-difference approach [5].

To compute such a family of steady states the numerical scheme would be mimetic and reproduce the behavior of the underlying system. The basic idea behind the mimetic finite-difference is to define difference operators that inherit the properties of differential operators (divergence, gradient, etc.), see for reference [7]. This approach permits the design of schemes that embody conservation laws and solution symmetries. It was shown in [6] that mimetic preservation of cosymmetry and some additional properties of nonlinear terms may be obtained using computer algebra system Maple.

Computation in three-dimensional Darcy convection problem is more difficult because the cosymmetry property does not exist for the three-dimensional case. However, using a special staggered grid approach [8] it was possible to construct

a mimetic discretization and to find the family of steady states for the Darcy convection in the parallelepiped. The key point was the approximation of nonlinear terms using the methodology outlined by Morinishi et. al. [9].

The multicomponent fluid in a porous medium significantly affects the convective flow [1]. In the two-dimensional case, again the cosymmetry property exists [10]. Using a mimetic mixed spectral and finite-difference approach which preserves the cosymmetry, convection of two- and three-component fluids for planar Darcy problem was analyzed in [11].

In this paper we study nontrivial strong nonuniqueness of convective patterns in a multicomponent fluid. We consider the three-dimensional problem of natural convection in a porous medium and develop a finite-difference scheme for a system in primitive variables (velocities, pressure, temperature, and mass fractions). The discretization is based on staggered grids approach as in [8] with five types of nodes for the variables of the problem. Using the differencing and averaging operators on two-nodes stencil we construct the special approximation of advective terms and check their conservation properties using computer algebra system Maple. The numerical verification of the scheme was done using Maple and MATLAB. The continuous family of planar patterns was computed when a depth of parallelepiped is small. A number of stable three-dimensional convective patterns are obtained in the case of rather large depth.

1 Darcy Convection Equations for Multicomponent Fluids

We consider a porous medium saturated by an incompressible multicomponent fluid which is heated from below. We assume that the Boussinesq approximation holds [1]; fluid velocities $\mathbf{v} = (v^1, v^2, v^3)^\top$ are assumed to be much smaller than the sound speed, so the fluid can be treated as incompressible. The density in the buoyancy term varies linearly with local temperature θ^1 and with mass fractions θ^r ($r = 2, \dots, S + 1$). The system of dimensionless equations [10] consists of the momentum equation based on the Darcy law

$$\varepsilon \frac{\partial \mathbf{v}}{\partial t} = -\nabla p - \mathbf{v} + \sum_{r=1}^{S+1} \lambda_r \theta^r \mathbf{k}, \tag{1}$$

the continuity equation

$$\nabla \cdot \mathbf{v} = 0, \tag{2}$$

and the equation for the deviation of temperature from linear (in z) profile θ^1 and deviation for each species θ^r ($r = 2, \dots, S + 1$)

$$b_r \frac{\partial \theta^r}{\partial t} + \mathbf{v} \cdot \nabla \theta^r = \kappa_r \Delta \theta^r + \mathbf{v} \cdot \mathbf{k}. \tag{3}$$

Here $\mathbf{k} = (0, 0, 1)^\top$ denotes the vector opposing to the direction of gravity, $p(x, y, z, t)$ is the pressure, and x, y, z are the space variables. Parameters

of relative porosity ε , Rayleigh numbers λ_r , and diffusion coefficients κ_r are given by

$$\varepsilon = \frac{K}{\mu l^2}, \quad \lambda_r = \frac{g\beta_r A_r l^2 K}{\nu^2}, \quad \kappa_r = \frac{\chi_r}{\nu}, \quad (4)$$

where K is the permeability coefficient, μ is the porosity of the medium, l is the length parameter, g is the gravity acceleration, β_r is the thermal expansion coefficient ($r = 1$) or the fractional expansion coefficient ($r > 1$), A_r is the characteristic temperature ($r = 1$) or concentration difference ($r > 1$), ν is viscosity, and χ_r is the thermal diffusivity of the fluid and mass fractions.

The parallelepiped $D = [0, L_x] \times [0, L_y] \times [0, L_z]$ with length L_x , depth L_y and height L_z is filled with the fluid. The normal component of the velocity is equal to zero at the boundary

$$\mathbf{v} \cdot \mathbf{n} = 0, \quad (x, y, z) \in \partial D. \quad (5)$$

We suppose that the temperature at the boundary is given by a linear function on the vertical coordinate z and consider the problem with mixed boundary conditions: the heat and concentration fluxes are equal to zero on two lateral faces $\partial_1 D = \{y = 0\} \cup \{y = L_y\}$, and the temperature deviation θ^r is equal to zero on the remaining faces $\partial_2 D = \partial D \setminus \partial_1 D$, see Fig. [1](#)

$$\theta_y^r = 0, \quad (x, y, z) \in \partial_1 D, \quad \theta^r = 0, \quad (x, y, z) \in \partial_2 D. \quad (6)$$

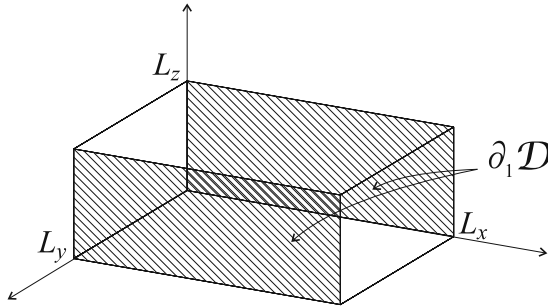


Fig. 1. Sketch of the boundary conditions

The initial condition is given as follows

$$\theta^r(x, y, z, 0) = \theta_0^r(x, y, z), \quad \mathbf{v}(x, y, z, 0) = \mathbf{v}_0(x, y, z). \quad (7)$$

It is easy to check that equations [\(1\)](#)–[\(6\)](#) are invariant with respect to the discrete symmetries

$$R_x : \{x, y, z, v^1, v^2, v^3, p, \theta^r\} \mapsto \{L_x - x, y, z, -v^1, v^2, v^3, p, \theta^r\}, \quad (8)$$

$$R_y : \{x, y, z, v^1, v^2, v^3, p, \theta^r\} \mapsto \{x, L_y - y, z, v^1, -v^2, v^3, p, \theta^r\}, \quad (9)$$

$$R_z : \{x, y, z, v^1, v^2, v^3, p, \theta^r\} \mapsto \{x, y, L_z - z, v^1, v^2, -v^3, p, -\theta^r\}, \quad (10)$$

where $r = 1, \dots, S + 1$. This implies the existence of solutions after appropriate transformations of velocity, pressure, and deviation of the temperature and mass fractions.

2 Staggered Grids

Equations (11)–(17) are discretized using five different types of nodes: one for the pressure, another for the temperature and concentrations, and three nodes for the components of velocity vector, see Fig. 2.

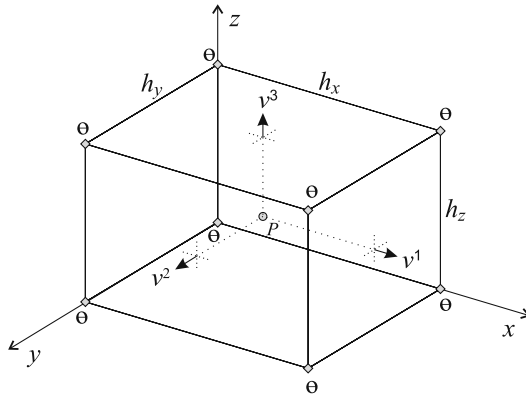


Fig. 2. A grid and nodes

Firstly we introduce the regular grids

$$\begin{aligned} x_i &= ih_x, \quad i = 0, \dots, N_x + 1, \quad h_x = L_x / (N_x + 1), \\ y_j &= -h_y/2 + jh_y, \quad j = 0, \dots, N_y + 1, \quad h_y = L_y / N_y, \\ z_k &= kh_z, \quad k = 0, \dots, N_z + 1, \quad h_z = L_z / (N_z + 1), \end{aligned}$$

and the staggered grids along all coordinates: $x_{i+1/2} = (x_i + x_{i+1})/2$, $i = 0, \dots, N_x$, $y_{j+1/2} = (y_j + y_{j+1})/2$, $j = 0, \dots, N_y$, $z_{k+1/2} = (z_k + z_{k+1})/2$, $k = 0, \dots, N_z$.

Thus, the temperature and mass fractions θ^r are defined at the nodes

$$\omega_0 = \{(x_i, y_j, z_k), i = 0, \dots, N_x + 1, j = 0, \dots, N_y + 1, k = 0, \dots, N_z + 1\}.$$

The velocities v^1 , v^2 and v^3 are defined on the grids which are staggered along the corresponding coordinates

$$\begin{aligned} \omega_1 &= \{(x_i, y_{j+1/2}, z_{k+1/2}), i = 0, \dots, N_x + 1, j = 0, \dots, N_y, k = 0, \dots, N_z\}, \\ \omega_2 &= \{(x_{i+1/2}, y_j, z_{k+1/2}), i = 0, \dots, N_x, j = 0, \dots, N_y + 1, k = 0, \dots, N_z\}, \\ \omega_3 &= \{(x_{i+1/2}, y_{j+1/2}, z_k), i = 0, \dots, N_x, j = 0, \dots, N_y, k = 0, \dots, N_z + 1\}. \end{aligned}$$

Finally, the pressure p is defined at the nodes

$$\omega_p = \{(x_{i+1/2}, y_{j+1/2}, z_{k+1/2}), i = 0, \dots, N_x, j = 0, \dots, N_y, k = 0, \dots, N_z\}.$$

The grids are introduced in such a way that at $\partial_2 D$, the boundary conditions for the temperature and mass fractions and for the normal component of velocity are fulfilled automatically. We define fictitious nodes for the temperature, concentration, and velocity v^2 to approximate the boundary conditions at $\partial_1 D$ with second-order accuracy.

2.1 Discrete Finite-Difference Operators

To approximate (II)–(VII) a set of discrete analogs of differential and averaging operators is defined on a two-point stencil

$$\begin{aligned} d_1 f_{i+1/2,j,k} &= \frac{f_{i+1,j,k} - f_{i,j,k}}{h_x}, & \delta_1 f_{i+1/2,j,k} &= \frac{f_{i+1,j,k} + f_{i,j,k}}{2} \\ d_2 f_{i,j+1/2,k} &= \frac{f_{i,j+1,k} - f_{i,j,k}}{h_y}, & \delta_2 f_{i,j+1/2,k} &= \frac{f_{i,j+1,k} + f_{i,j,k}}{2} \\ d_3 f_{i,j,k+1/2} &= \frac{f_{i,j,k+1} - f_{i,j,k}}{h_z}, & \delta_3 f_{i,j,k+1/2} &= \frac{f_{i,j,k+1} + f_{i,j,k}}{2}. \end{aligned} \tag{11}$$

Formulas (II) are valid both for integer and half-integer values of i , j , and k . Then the discrete analog of the Laplacian on the seven-nodes stencil can be written as

$$\Delta_h = d_1 d_1 + d_2 d_2 + d_3 d_3 \approx \Delta, \tag{12}$$

and the averaging operator on three-dimensional cell is given as

$$\delta_0 = \delta_1 \delta_2 \delta_3. \tag{13}$$

The construction of the nonlinear term approximation is done using a linear combination of two terms

$$\begin{aligned} (\mathbf{v} \cdot \nabla f)_{i,j,k} &\approx J(f, \mathbf{v})_{i,j,k} \\ &= \left[\alpha \sum_{s=1}^3 d_s \delta_s \left(f \prod_{n \neq s}^3 \delta_n v^s \right) + (1 - \alpha) \sum_{s=1}^3 d_s \prod_{n \neq s}^3 \delta_n (\delta_0 f \delta_s v^s) \right]_{i,j,k} \end{aligned} \tag{14}$$

To find suitable parameter values α we apply Maple. It has been found that the value $\alpha = 1/3$ constitutes mimetic discretization of the underlying problem. Then we verify this conclusion via direct numerical experiment with computations of the planar convective regimes and analyze their belongings to the family of steady states.

2.2 Semi-discretization

To compute some steady states we use an artificial compressibility and consider the equation with coefficient ζ

$$\partial_t p + \zeta^{-1} \nabla \cdot \mathbf{v} = 0 \tag{15}$$

instead of equation (2).

Using the operators (11)–(14), the system (1), (3), and (15) is discretized in the following form

$$\left[b^r \dot{\theta}^r - \kappa_r \Delta_h \theta^r - \delta_1 \delta_2 v^3 + J(\theta^r, v) \right]_{i,j,k} = 0, \quad r = 1, \dots, S + 1, \tag{16}$$

$$\left[\varepsilon v^1 + d_1 p + v^1 \right]_{i,j+1/2,k+1/2} = 0, \tag{17}$$

$$\left[\varepsilon v^2 + d_2 p + v^2 \right]_{i+1/2,j,k+1/2} = 0, \tag{18}$$

$$\left[\varepsilon v^3 + d_3 p + v^3 - \sum_{r=1}^{S+1} \lambda_r \delta_1 \delta_2 \theta^r \right]_{i+1/2,j+1/2,k} = 0, \tag{19}$$

$$\left[\zeta \dot{p} + d_1 v^1 + d_2 v^2 + d_3 v^3 \right]_{i+1/2,j+1/2,k+1/2} = 0. \tag{20}$$

The problem (1)–(6) is discretized using fictitious nodes to satisfy the boundary conditions on the planes $y = 0$ and $y = L_y$. The discretization of the boundary conditions is given below as:

- for $x = 0$ ($i = 0$) and $x = L_x$ ($i = N_x + 1$):

$$\begin{aligned} v_{i,j+1/2,k+1/2}^1 &= 0, \quad j = 0, \dots, N_y, \quad k = 0, \dots, N_z, \\ \theta_{i,j,k}^r &= 0, \quad j = 0, \dots, N_y + 1, \quad k = 0, \dots, N_z + 1, \end{aligned} \tag{21}$$

- for $y = 0$ ($j = 0$) and $y = L_y$ ($j = N_y + 1$):

$$\begin{aligned} v_{i+1/2,0,k+1/2}^2 &= -v_{i+1/2,1,k+1/2}^2, \quad i = 0, \dots, N_x, \quad k = 0, \dots, N_z, \\ v_{i+1/2,N_y+1,k+1/2}^2 &= -v_{i+1/2,N_y,k+1/2}^2, \quad i = 0, \dots, N_x, \quad k = 0, \dots, N_z, \\ \theta_{i,0,k}^r &= \theta_{i,1,k}^r, \quad \theta_{i,N_y+1,k}^r = \theta_{i,N_y,k}^r, \quad i = 0, \dots, N_x + 1, \quad k = 0, \dots, N_z + 1. \end{aligned} \tag{22}$$

- for $z = 0$ ($k = 0$) and $z = L_z$ ($k = N_z + 1$):

$$\begin{aligned} v_{i+1/2,j+1/2,k}^3 &= 0, \quad i = 0, \dots, N_x, \quad j = 0, \dots, N_y, \\ \theta_{i,j,k}^r &= 0, \quad i = 0, \dots, N_x + 1, \quad j = 0, \dots, N_y + 1. \end{aligned} \tag{23}$$

2.3 Computational Procedure

We rewrite the resulting system of equations (17)–(21) in vector form by introducing vectors which contain only unknowns at internal nodes

$$\Theta^r = (\theta_{111}^r, \dots, \theta_{N_x 11}^r, \theta_{121}^r, \dots, \theta_{N_x N_y N_z}^r),$$

$$\begin{aligned} V^1 &= (v_{111}^1, \dots, v_{N_x 11}^1, v_{121}^1, \dots, v_{N_x(N_y+1)(N_z+1)}^1), \\ V^2 &= (v_{111}^2, \dots, v_{N_x+1,11}^2, v_{121}^2, \dots, v_{(N_x+1)N_y(N_z+1)}^2), \\ V^3 &= (v_{111}^3, \dots, v_{N_x+1,11}^3, v_{121}^3, \dots, v_{(N_x+1)(N_y+1)N_z}^3), \\ P &= (p_{111}, \dots, p_{N_x+1,11}, p_{121}, \dots, p_{(N_x+1)(N_y+1)(N_z+1)}), \end{aligned}$$

and obtain the system with $V = (V^1, V^2, V^3)^T$

$$\begin{aligned} \dot{\Theta}^r &= \kappa_r A_1 \Theta^r + C_1 V^3 - J(\Theta^r, V), \quad r = 1, \dots, S + 1, \\ \dot{V}^k &= -B_{3+k} P - C_{1+k} V^k + \delta_{3k} \sum_{r=1}^{S+1} \lambda_r C_5 \Theta^r, \\ \dot{P} &= -\sum_{k=1}^3 B_k V^k, \quad k = 1, 2, 3. \end{aligned} \tag{24}$$

Here the matrices $B_k, k = 1, \dots, 6$, are constructed by the first-order difference operators (11), and the matrices $C_k, k = 1, \dots, 5$, are constructed by the averaging operators (12). The matrix A_1 represents the discrete form of the Laplacian. The nonlinear term is given by $J(\Theta^r, V)$. The number of unknowns in the system of equations (24) is

$$(5 + S)N_x N_y N_z + 3(N_x N_y + N_x N_z + N_y N_z) + 2(N_x + N_y + N_z) + 1.$$

From (24) at $J = 0$ we can derive the perturbation equations (σ is a decrement of linear growth) to analyze the stability of the state of rest

$$\sigma \Theta^r = \kappa_r A_1 \Theta^r + C_1 V^3, \quad \sigma P = -\sum_{k=1}^3 B_k V^k, \tag{25}$$

$$\sigma V^k = -B_{3+k} P - C_{1+k} V^k + \delta_{3k} \sum_{r=1}^{S+1} \lambda_r C_5 \Theta^r, \quad k = 1, 2, 3. \tag{26}$$

For the decrement $\sigma = 0$ we obtain the system from which we can determine the threshold value of the Rayleigh number corresponding to the monotonic loss of stability. We can express P, V^1, V^2, V^3 via Θ^r from (25)–(26) and obtain a system of $N_x N_y N_z$ equations for the unknown vector Θ^r

$$\kappa_r A_1 \Theta^r = C_1 (C_5 - B_6 Q) \sum_{s=1}^{S+1} \lambda_s \Theta^s, \quad r = 1, \dots, S + 1. \tag{27}$$

Here we find the vector $P = Q \sum_{r=1}^{S+1} \lambda_r \Theta^r$ from the system of rank one deficient linear algebraic equations

$$\sum_{k=1}^3 B_k C_{1+k}^{-1} B_{3+k} P = B_3 C_4^{-1} C_5 \sum_{r=1}^{S+1} \lambda_r \Theta^r.$$

Since for an incompressible flow the pressure may differ by a constant we can exclude one component of P and one respective equation.

To compute a family of steady states we apply the technique based on the cosymmetric version of the implicit function theorem [3]. It contains numerical computation of the matrix of linearization for a number of steady states, application of the SVD technique, and the continuation method. To verify this approach we carried out a number of experiments using Maple and MATLAB. Numerical computation of a matrix of linearization in MATLAB was compared with corresponding analytical derivation in Maple. The tests with several hundreds of nodes proved the possibility of the direct numerical approach. Similarly the computation of a kernel for the matrix of linearization was found rather robust, the corresponding spectral value was about 10^{-7} . It was enough for the determination of direction along the family of steady states with admissible accuracy. To find an isolated convective pattern we apply the direct approach and integrate the system of ordinary differential equations (24) by the classical fourth-order Runge–Kutta method up to convergence.

3 Numerical Results

We give some computational results of convective flows of two-component fluid ($S = 1$) in the parallelepiped with $L_x = 3$, $L_z = 1$ for several values of the

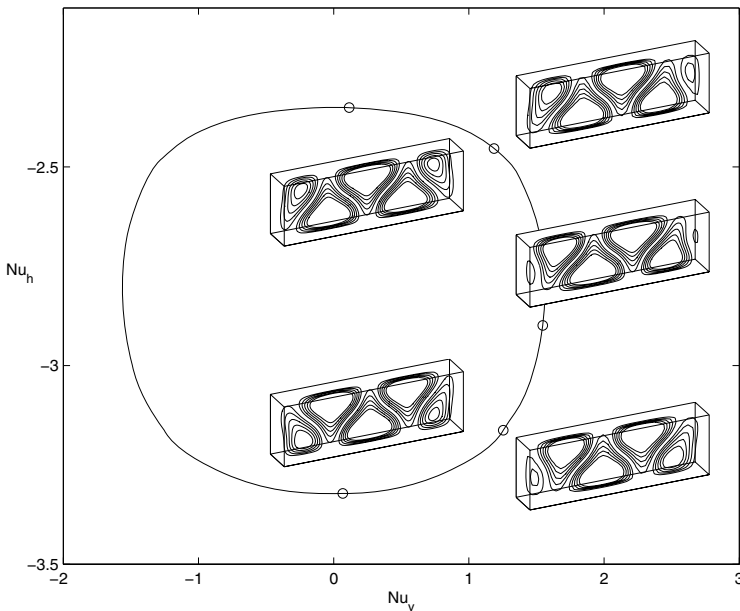


Fig. 3. Planar regimes from the family of steady states

depth L_y . The problems were analyzed for Rayleigh numbers $\lambda_1 = 120$, $\lambda_2 = 10$, and for the coefficients $k_1 = 1$, $k_2 = 0.2$, $b_1 = b_2 = 1$, $\varepsilon = 0.05$. The computations of the discretized equations were performed for the grids with $60 \times 10 \times 20$ and $60 \times 20 \times 20$ internal nodes. When the depth L_y was rather small we have detected only planar convective flows which illustrates the cosymmetry phenomena: the state of rest lost its stability, and a one-parameter family of steady states branched off. Several steady regimes from the family are presented in Fig. 3, where the depth of parallelepiped was $L_y = 0.5$. Because all the states are planar we give only slice in $x - z$ plane. One can see that convective patterns continuously transform when we move along the curve of the family.

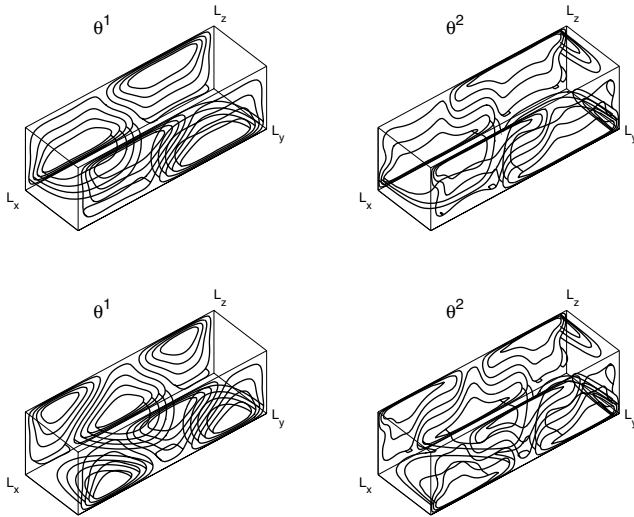


Fig. 4. Two convective regimes with the same number of convective cells on the planes $y = 0$ and $y = L_y$

There exists a critical value of the depth L_y^* such that for $L_y > L_y^*$ a fixed number of isolated convective patterns occurs after loss of stability of the state of rest. For $L_y > L_y^*$, a family may exist only as a set of unstable planar regimes. Depending on initial conditions we have found different isolated three-dimensional regimes in the parallelepiped with depth $L_y = 1$, see Figs. 4–6. First two regimes are presented by its distribution of temperature θ^1 and mass fraction θ^2 in Fig. 4. Upper (lower) ones are characterized by three (four) convective cells on the planes $y = 0$ and $y = L_y$. Because of discrete symmetries in this problem there exist similar flows which are given by corresponding reflections.

We have also observed more complicated regimes with different structure of the temperature (mass fraction) on the planes $y = 0$ and $y = L_y$. Distributions of the temperature in different sections are shown in Fig. 5, where the discrete symmetry is also observed in this flow.

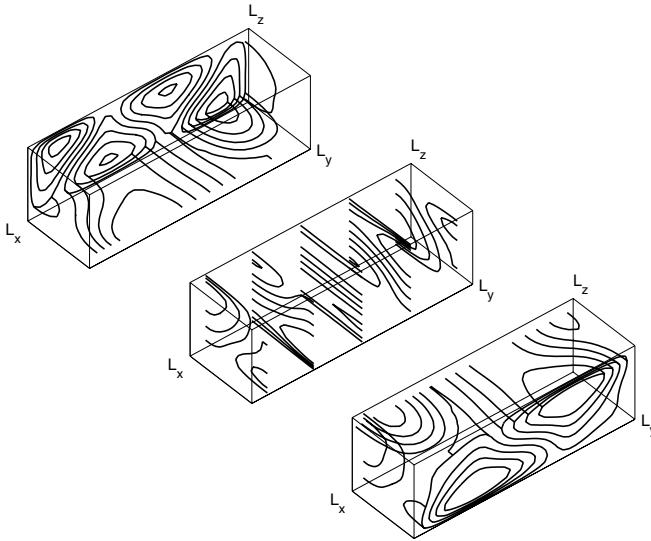


Fig. 5. Convective pattern with different numbers of convective cells on the planes $y = 0$ and $y = L_y$

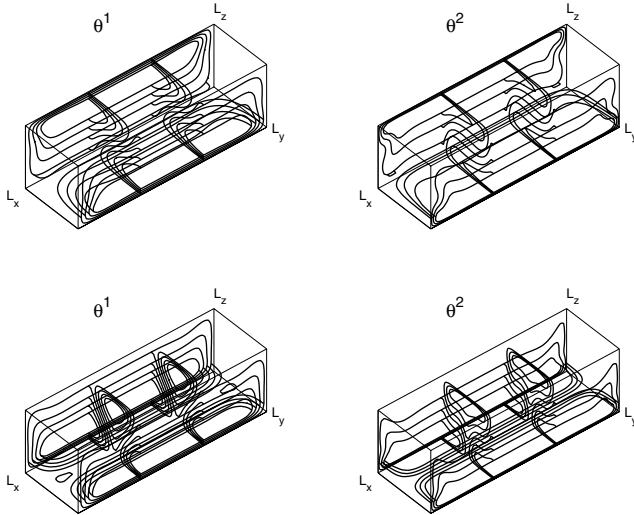


Fig. 6. Convective flows close to the planar regimes if x is far from the planes $x = 0$ and $x = L_x$

It is interesting that for the given parallelepiped the most stable regimes appear two-dimensional (constant in direction x), which is demonstrated in Fig. 6, where three-dimensional character of the flow is only revealed in the vicinity the planes $x = 0$ and $x = L_x$. This is the case when length L_x is significantly greater than depth and height.

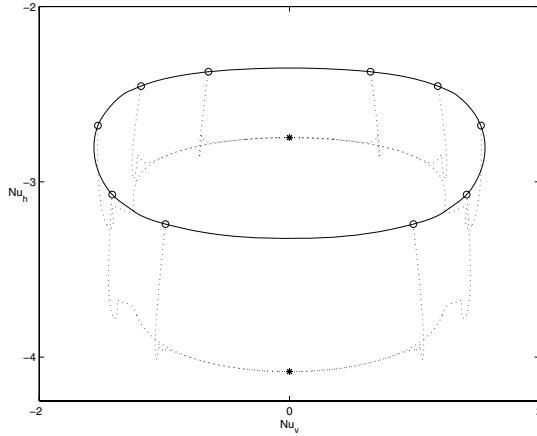


Fig. 7. Destruction of the family of steady states under non-mimetic approximation of advective terms

Finally we demonstrate that the non-mimetic approximation destroys the family of steady states. When simpler approximation

$$(\mathbf{v} \cdot \nabla f)_{i,j,k} \approx \left[\sum_{s=1}^3 d_s \delta_s f \prod_{n \neq s} \delta_n v^n \right]_{i,j,k} . \tag{28}$$

instead of (15) was used, only two convective flows are obtained through computations. Figure 7 depicts that from different initial states (circles) we come to symmetric patterns (marked by stars). Convergence may take some time but the typical trajectory (dotted line) reproduces the movement along the 'lost' family (solid line). One can see it as a shadow of the missing object.

Summary

Using a mimetic scheme on staggered grids for a three-dimensional convective multicomponent flow in a porous medium, the preservation of the cosymmetry and discrete symmetries were demonstrated. It was also shown that a non-mimetic scheme can destroy the family of steady states.

Acknowledgements

V.T. was partially supported by the Program 'Scientific potential development' by Russian Ministry for Education and Research, project 2.1.1/6095.

References

1. Nield, D.A., Bejan, A.: Convection in porous media. Springer, New York (1999)
2. Lyubimov, D.V.: On the convective flows in the porous medium heated from below. *J. Appl. Mech. Techn. Phys.* 16, 257–261 (1975)
3. Yudovich, V.I.: Cosymmetry, degeneracy of the solutions of operator equations, and the onset of filtrational convection. *Math. Notes* 49, 540–545 (1991)
4. Govorukhin, V.N.: Numerical simulation of the loss of stability for secondary steady regimes in the Darcy plane-convection problem. *Doklady Akademii Nauk* 363, 806–808 (1998)
5. Karasözen, B., Tsybulin, V.G.: Finite-difference approximation and cosymmetry conservation in filtration convection problem. *Physics Letters A* 262, 321–329 (1999)
6. Karasözen, B., Tsybulin, V.G.: Conservative finite difference schemes for cosymmetric systems. In: *Proc. 4th Conf. on Computer Algebra in Scientific Computing*, pp. 363–375. Springer, Heidelberg (2001)
7. Hyman, J.M., Bochev, P.B.: Principles of Mimetic Discretizations of Differential Operators. *IMA Volumes in Mathematics and Its Applications* 142, 89–114 (2006)
8. Karasözen, B., Nemtsev, A.D., Tsybulin, V.G.: Staggered grids discretization in three-dimensional Darcy convection. *Comput. Phys. Comm.* 170, 885–893 (2008)
9. Morinishi, Y., Lund, T.S., Vasilyev, O.V., Moin, P.: Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow. *J. Comput. Phys.* 143, 90–124 (1998)
10. Yudovich, V.I.: Cosymmetry and convection of a multicomponent fluid in a porous medium. *Izv. Vuzov. Severo-Kavkazsk. Region. Estestv. Nauki. Spetsvypusk. Mat. Modelirovanie*, 174–178 (2001)
11. Kantur, O.Y., Tsybulin, V.G.: Numerical investigation of the plane problem of convection of a multicomponent fluid in a porous medium. *Fluid Dynamics* 39, 464–473 (2004)

Symbolic-Numerical Algorithms for Solving Parabolic Quantum Well Problem with Hydrogen-Like Impurity

S.I. Vinitsky, O. Chuluunbaatar, V.P. Gerdt, A.A. Gusev, and V.A. Rostovtsev

Joint Institute for Nuclear Research, Dubna, Russia
vinitsky@theor.jinr.ru

Abstract. For parabolic quantum well problem with hydrogen-like impurity a two-dimensional boundary-value problem is formulated in spherical coordinates at fixed magnetic quantum number. Computational scheme using *modified* angular prolate spheroidal functions is presented. Symbolic-numerical algorithms for solving the problem are elaborated. The efficiency of the algorithms and their implementation is demonstrated by solving typical test examples and proving the compatibility conditions for asymptotic solutions of scattering problems in spherical and cylindrical coordinates.

Keywords: Symbolic-numerical algorithms, parabolic quantum well, hydrogen-like impurity, modified prolate angular spheroidal functions.

1 Introduction

In [1] optical absorption into the ground state of GaAs parabolic quantum well and rectangular quantum well with infinitely high walls in the presence of a hydrogen-like impurity was considered. Calculation of the ground state of these quantum wells was carried out using single-parameter variational functions in the cylindrical coordinate system. The upper bounds of these energies were obtained depending on the shift of the Coulomb potential center. The analysis of more complex quantum mechanical models leads to boundary-value problems in a non-standard domain of the configuration space with complex boundary, solved using finite-element method [2,3], or by means of reducing the problem to ordinary differential equations following Kantorovich method [4], known in physics as the adiabatic approach to quantum mechanical problems with slow and fast variables. In the Kantorovich method, the basis functions depend upon the slow variables as parameters and obey the boundary conditions that account for all specific features of the original problem. This provides the efficiency of the method for solving boundary-value problems in a non-standard domain, e.g., in a sector of a circle with mixed boundary conditions [5], as well as in the presence of singular potential against the background of confining potentials of the oscillator type with respect to some independent variables [6,7]. The latter determines the potentialities of using the method to analyze low-dimensional quantum mechanical models of semiconductor nanostructures [8].

In this paper we present a scheme for solving the boundary-value problem for a parabolic quantum well in the adiabatic representation and in the spherical coordinates. For efficient application of the Kantorovich method we elaborated the following symbolic-numerical algorithms to compute the appropriate quantities to a prescribed accuracy:

- numerical solution of the parametric self-adjointed Sturm-Liouville problem on a bounded interval of the parameter values and calculation of derivatives with respect to the parameter of the eigenfunctions and of the matrix elements (integrals of the eigenfunctions multiplied by their derivatives with respect to the parameter) that appear as variable coefficients in the system of second-order ordinary differential equations (ODPEVP, implemented in FORTRAN [9]),
- asymptotic forms of the eigenfunctions and of the matrix elements that appear as variable coefficients in asymptotic solutions of the boundary-value problem under consideration and in the asymptotic forms of the system of second-order ordinary differential equations (MATRA, implemented in MAPLE),
- asymptotic forms of the solutions of the system of second-order ordinary differential equations for small and large values of the radial variable needed for solving the corresponding boundary-value problem with the third-type boundary conditions (ASYMRS, implemented in MAPLE),
- numerical solutions of the boundary-value problem for a system of second-order ordinary differential equations (KANTBP, implemented in FORTRAN [5]).

The paper is organized as follows. In Section 2, the statement of the boundary-value problem is given. In Section 3, the procedure MATRA for analytic calculation of asymptotic form of basis functions and matrix elements at large values of the radial variable is described. In Section 4, the procedure ASYMRS for the calculation of asymptotic forms of fundamental solutions of a system of radial equations at large values of radial variable in the analytic form is presented. In Section 5, a test example of numerical calculation of the ground state energy and wave functions with the help of ODPEVP and KANTBP programs is given. The Conclusion outlines further applications of the above set of symbolic-numerical algorithms and programs.

2 Problem Statement

The Schrödinger equation describing the parabolic quantum well problem with shifted hydrogen-like impurity in the reduced atomic units and in the spherical coordinates $(r, \eta = \cos \theta, \phi)$ at a fixed magnetic quantum number m reads as [4]

$$\left(-\frac{1}{r^2} \frac{\partial}{\partial r} r^2 \frac{\partial}{\partial r} + \frac{1}{r^2} A(c, b) - \frac{2q}{r} \right) \psi_m(r, \eta) = 2E \psi_m(r, \eta). \quad (1)$$

Here $A(c, b) \equiv A^{(0)}(c, b) + c^2 + f$ is the operator of the modified angular functions, which at $b = f = 0$ correspond to the angular prolate spheroidal functions [10]

$$A^{(0)}(c, b) = -\frac{\partial}{\partial \eta} (1 - \eta^2) \frac{\partial}{\partial \eta} + \frac{m^2}{1 - \eta^2} + c^2 (\eta^2 - 1) - b\eta, \quad (2)$$

where $c = \omega r^2$, $b = -2\omega^2 z_c r^3$, and $f = (\omega z_c r)^2$ are real parameters depending on the harmonic oscillator frequency ω and the shift z_c of the Coulomb charge q along z -axis from the origin of the cylindrical frame (ρ, z, ϕ) in \mathbf{R}^3 , i.e., $r = \sqrt{\rho^2 + (z - z_c)^2}$. The wave functions $\psi_m(r, \eta, b) \equiv \psi_{mi}(r, \eta, b) \equiv \psi_{mi}(r, \eta, z_c)$ at fixed m obey the following conditions at the boundary of the domain $\Omega_{r, \eta} = \Omega(0 \leq r < \infty, -1 \leq \eta \leq 1)$:

$$\lim_{\eta \rightarrow \pm 1} (1 - \eta^2) \frac{\partial \psi_m(r, \eta)}{\partial \eta} = 0, \text{ for } m = 0, \text{ and } \psi_m(r, \pm 1) = 0, \text{ for } m \neq 0,$$

$$\lim_{r \rightarrow 0} r^2 \frac{\partial \psi_m(r, \eta)}{\partial r} = 0.$$

At large $r = r_{\max} \gg 1$ the discrete-spectrum wave functions obey the Dirichlet boundary condition that follows from the asymptotic behavior of the solution

$$\lim_{r \rightarrow +\infty} r^2 \psi_m(r, \eta) = 0 \quad \rightarrow \quad \psi_m(r_{\max}, \eta) = 0,$$

and also the orthonormality condition

$$\int_0^{r_{\max}} \int_{-1}^1 \psi_{mi}(r, \eta) \psi_{mj}(r, \eta) r^2 dr d\eta = \delta_{ij}. \tag{3}$$

The solution of (1)–(3) at fixed m is sought in the form of the Kantorovich expansion with respect to the single-parameter functions $\Phi_j(\eta; r) \equiv \Phi_{mj}(\eta; r)$:

$$\psi_{mi}(r, \eta) = \sum_{j=1}^{j_{\max}} \Phi_{mj}(\eta; r) \chi_{ji}(r), \tag{4}$$

Here the functions $\chi_{ji}(r)$ are to be found, while the basis functions $\Phi_j(\eta; r) \in F_r \sim L_2[-1, 1]$ are solutions of the eigenvalue problem:

$$A(c, b) \Phi_{mj}(\eta; r) = E_j(r) \Phi_{mj}(\eta; r). \tag{5}$$

The eigenfunctions $\Phi_{mj}(\eta; r) \equiv \Phi_{mj}(r, \eta, z_c)$ at fixed m obey the symmetry condition $\Phi_{mj}(r, \eta, z_c) = \exp(i\pi\nu_{mq}) \Phi_{mj}(r, -\eta, -z_c)$, where $\nu_{mq} \equiv \nu_{mq}(r, z_c)$ is the real phase, q is the number of zeros in $\eta \in [-1, 1]$, in particular, $\nu_{mq}(r, 0) = q$ at $z_c = 0$, $E_j(r, z_c) = E_j(r, -z_c)$, and the boundary conditions with respect to the angular variable η at each fixed value of the parameter $r \in \mathbf{R}_+^1$

$$\lim_{\eta \rightarrow \pm 1} (1 - \eta^2) \frac{\partial \Phi_{mj}(\eta; r)}{\partial \eta} = 0, \text{ for } m = 0, \text{ and } \Phi_{mj}(r, \pm 1) = 0, \text{ for } m \neq 0, \tag{6}$$

as well as the orthonormality conditions in the interval $\Omega_\eta = [-1, 1]$:

$$\langle \Phi_{mi}(\eta; r) | \Phi_{mj}(\eta; r) \rangle_{\Omega_\eta} = \int_{-1}^1 \Phi_{mi}(\eta; r) \Phi_{mj}(\eta; r) d\eta = \delta_{ij}. \tag{7}$$

Note that the eigenvalues $E_j(r)$ of the operator $A(c, b)$ from (5) are related to the eigenvalues $\lambda_j(r)$ of the operator $A^{(0)}(c, b)$ from (2) by the equality

$E_j(r) = \lambda_j(r) + c^2 + f$. The projection of Eq. (10), using expansion (4), is reduced to the set of j_{\max} ordinary second-order differential equations with respect to the unknown vector function $\chi^{(i)}(r) \equiv (\chi_{1i}(r), \dots, \chi_{j_{\max}i}(r))$:

$$\left(-\frac{1}{r^{d-1}} \mathbf{I} \frac{d}{dr} r^{d-1} \frac{d}{dr} + \frac{\mathbf{U}(r)}{r^2} + \mathbf{Q}(r) \frac{d}{dr} + \frac{1}{r^{d-1}} \frac{dr^{d-1} \mathbf{Q}(r)}{dr} - 2E \mathbf{I} \right) \chi^{(i)}(r) = 0. \tag{8}$$

Here $d = 3$ is the dimension of the above space \mathbf{R}^3 , \mathbf{I} , $\mathbf{U}(r)$, and $\mathbf{Q}(r)$ are $j_{\max} \times j_{\max}$ matrices whose entries are defined by the following relations:

$$\begin{aligned} U_{ij}(r) &= r^2 H_{ij}(r) + \frac{E_i(r) + E_j(r)}{2} \delta_{ij} - 2qr \delta_{ij} \quad I_{ij} = \delta_{ij}, \\ H_{ij}(r) &= H_{ji}(r) = \left\langle \frac{\partial \Phi_i(\eta; r)}{\partial r} \left| \frac{\partial \Phi_j(\eta; r)}{\partial r} \right\rangle_{\Omega_\eta}, \\ Q_{ij}(r) &= -Q_{ji}(r) = - \left\langle \Phi_i(\eta; r) \left| \frac{\partial \Phi_j(\eta; r)}{\partial r} \right\rangle_{\Omega_\eta}. \end{aligned} \tag{9}$$

The discrete-spectrum solutions obey the asymptotic boundary conditions and the orthonormality condition

$$\lim_{r \rightarrow 0} r^{d-1} \frac{d\chi^{(i)}(r)}{dr} = 0, \quad \lim_{r \rightarrow \infty} r^{d-1} \chi^{(i)}(r) = 0 \quad \rightarrow \quad \chi^{(i)}(r_{\max}) = 0, \tag{10}$$

$$\int_0^{r_{\max}} r^{d-1} \left(\chi^{(i)}(r) \right)^T \chi^{(j)}(r) dr = \delta_{ij}. \tag{11}$$

Remark 1. The continuity of the eigenfunction $\Phi_j(\eta; r)$ with respect to the parameter r is very important for calculations of the potential matrix elements (9) and their further applications for solution of boundary problems for a system of coupled differential equations (8) as considered in [5]. Hence we required $\Phi_j(\eta; r) > 0$ in the vicinity of the right boundary point $\eta = 1$ [9].

Remark 2. The formulation of the boundary-value problem of continuous spectrum for the set of Eqs. (8) using asymptotic expansions of the solutions presented below is given in [5,7].

3 Symbolic Algorithm for Evaluating the Asymptotic Forms of Matrix Elements

The procedure MATRA computes the asymptotic forms of solutions of the eigenvalue problem (5) together with the matrix elements (9) as expansions in powers of r and $1/r$ for small and large values of r , respectively. Here we consider the case of large r .

In **step 1** we go from the coordinate $\eta \in [-1, 1]$ to the new coordinate $\hat{z} \in [\sqrt{\omega}(-r + z_c), \sqrt{\omega}(r + z_c)]$ using the formula $r\eta \equiv z' = z - z_c = (\hat{z} - z_c \sqrt{\omega}) / \sqrt{\omega}$.

In **step 2** we construct the asymptotic expansion defined in the domain $\eta \in [-\eta_1, \eta_1]$, where $\eta_1 = O((\omega r^2)^{-1/2+\epsilon})$, $0 < \epsilon < 1/2$. It means that in

the evaluation of the corresponding integrals we omit exponentially small terms and change the domain from the finite interval $[\sqrt{\omega}(-r + z_c), \sqrt{\omega}(r + z_c)]$ to the infinite one $(-\infty, +\infty)$.

In **step 3** we find the asymptotic solution $\Phi_j^{as}(\hat{z}; r)$ and $r^{-2}E_j(r) = r^{-2}(\lambda_j(r) + c^2 + f) = \omega\beta_j(r)$ as an expansion with $j = n + 1$

$$\Phi_j^{as}(\hat{z}; r) = \sqrt[4]{\omega}\sqrt{r} \sum_{k=0}^{k_{\max}} \frac{\Phi_n^{(2k)}(\hat{z})}{r^{2k}}, \quad E_j(r) = \sum_{k=0}^{k_{\max}} \frac{E_n^{(2k)}}{r^{2k}}, \quad \beta_j(r) = \sum_{k=0}^{k_{\max}} \frac{\beta_n^{(2k)}}{r^{2k}}. \quad (12)$$

Substituting Eq. (12) into Eq. (5) and equating the coefficients at the same powers of r , we arrive at a system of recurrence differential equations for evaluating the coefficients $\Phi_n^{(2k)}(\hat{z})$ and $\beta_n^{(2k)}$, $k = 1, \dots, k_{\max}$:

$$L(n)\Phi_n^{(2k)} = f_n^{(2k)}(\hat{z}), \quad L(n) = -\frac{d^2}{d\hat{z}^2} - (2n + 1) + \hat{z}^2, \quad (13)$$

with the initial data $\beta_n^{(0)} = 2n + 1$, and $\Phi_n^{(0)}(\hat{z})$ is a known solution of the problem

$$L(n)\Phi_n^{(0)}(\hat{z}) = 0, \quad \int_{-\infty}^{+\infty} \Phi_n^{(0)}(\hat{z})\Phi_{n'}^{(0)}(\hat{z})d\hat{z} = \delta_{nn'}. \quad (14)$$

In Eqs. (13) the right-hand sides $f_n^{(2k)}(\hat{z})$ are defined by the relations

$$f_n^{(2k)}(\hat{z}) = \frac{(\hat{z} - z_c\sqrt{\omega})^2}{\omega} \frac{d^2\Phi_n^{(2k-2)}(\hat{z})}{d\hat{z}^2} + \frac{2(\hat{z} - z_c\sqrt{\omega})}{\omega} \frac{d\Phi_n^{(2k-2)}(\hat{z})}{d\hat{z}} + \sum_{j=1}^k (a^{(2j)}(\hat{z}, z_c) - \beta_n^{(2j)})\Phi^{(2k-2j)}(\hat{z}) = 0,$$

where the coefficients $a^{(2j)}(\hat{z}, z_c)$ are defined by Taylor expansion at large ωr^2

$$\frac{m^2}{1 - \eta^2} = m^2 \left(1 - \frac{(\hat{z} - z_c\sqrt{\omega})^2}{\omega r^2} \right)^{-1} = \sum_{j=0}^{k_{\max}} \frac{a^{(2j)}(\hat{z}, z_c)}{r^{2j}}. \quad (15)$$

Note that the coefficients $a^{(2j)}(\hat{z}, z_c)$ contain the terms of the order of \hat{z}^{2l} till $l = j$. The orthogonality and normalization conditions follow from (7) and (12)

$$I_{jj'}^{(2k)} = \sum_{l=0}^k \int_{-\infty}^{+\infty} \Phi_{n_l}^{(2l)}(\hat{z})\Phi_{n_r}^{(2k-2l)}(\hat{z})d\hat{z} = \delta_{k0}\delta_{n_l n_r}, \quad (16)$$

where $n_l = j - 1$, $n_r = j' - 1$.

We find the asymptotic expressions of the matrix elements $H_{jj'}(r)$ and $Q_{jj'}(r)$ from (9) in the form of expansions

$$Q_{jj'}(r) = \sum_{k=1}^{k_{\max}} \frac{Q_{jj'}^{(2k-1)}}{r^{2k-1}}, \quad H_{jj'}(r) = \sum_{k=1}^{k_{\max}} \frac{H_{jj'}^{(2k)}}{r^{2k}}. \quad (17)$$

Table 1. Values of the partial sums (27) for $r^{-2}E_j(r)$ depending on k_{\max} for $\omega = 3$, $m = 0$, $z_c = 0.4$, and $r = 8$. The last row contains the corresponding numerical values (n.v.) calculated by means of ODPEVP [9].

j	$r^{-2}E_1$	$r^{-2}E_2$	$r^{-2}E_3$	$r^{-2}E_4$
$r^{-0}E_j^{(0)}$	3	9	15.	21.
$+r^{-2}E_j^{(2)}$	2.9845312	8.9614062	14.922656	20.868281
$+r^{-4}E_j^{(4)}$	2.9844843	8.9611764	14.922034	20.866998
$+r^{-6}E_j^{(6)}$	2.9844838	8.9611729	14.922022	20.866966
$+r^{-8}E_j^{(8)}$	2.9844838	8.9611729	14.922022	20.866965
$+r^{-10}E_j^{(10)}$	2.9844838	8.9611729	14.922022	20.866965
(n.v.)	2.9844838	8.9611729	14.922022	20.866965

Here the coefficients $Q_{jj'}^{(2k+1)}$ and $H_{jj'}^{(2k+2)}$ are defined by the relations

$$\begin{aligned}
 Q_{jj'}^{(2k+1)} &= -\sum_{l=0}^k \int_{-\infty}^{+\infty} \Phi_{n_l}^{(2l)}(\hat{z}) \hat{Q} \Phi_{n_r}^{(2k-2l)}(\hat{z}) d\hat{z}, \\
 H_{jj'}^{(2k+2)} &= \sum_{l=0}^k \int_{-\infty}^{+\infty} \hat{Q} \Phi_{n_l}^{(2l)}(\hat{z}) \hat{Q} \Phi_{n_r}^{(2k-2l)}(\hat{z}) d\hat{z}, \\
 \hat{Q} \Phi_{n_l}^{(2l)}(\hat{z}) &= \left(\frac{1}{2} - 2l\right) \Phi_{n_l}^{(2l)}(\hat{z}) + (\hat{z} - z_c \sqrt{\omega}) \frac{d\Phi_{n_l}^{(2l)}(\hat{z})}{d\hat{z}}.
 \end{aligned}
 \tag{18}$$

In step 4 we construct $\Phi_n^{(2k)}(\hat{z})$ as the expansion with unknown coefficients $b_{n;s}^{(2k)}$

$$\Phi_n^{(2k)}(\hat{z}) = \sum_{s=-M(k)}^{M(k)} b_{n;s}^{(2k)} \Phi_{n+s}^{(0)}(\hat{z}).
 \tag{19}$$

Here the basis functions $\Phi_v^{(0)}(\hat{z})$ are solutions of (14) expressed in terms of the Hermite polynomials [10]

$$\Phi_v^{(0)}(\hat{z}) = \frac{H_v(\hat{z}) \exp(-\hat{z}^2/2)}{\sqrt[4]{\pi} \sqrt{2^v} \sqrt{v!}}.$$

Using the known recurrence relation for Hermite polynomials $H_v(\hat{z})$

$$\hat{z}H_v(\hat{z}) = \frac{H_{v+1}(\hat{z})}{2} + vH_{v-1}(\hat{z}), \quad \frac{dH_v(\hat{z})}{d\hat{z}} = 2vH_{v-1}(\hat{z}),
 \tag{20}$$

we obtain the recurrence relations for the basis functions $\Phi_v^{(0)}(\hat{z})$:

$$\begin{aligned}
 \hat{z}\Phi_v^{(0)}(\hat{z}) &= +\frac{\sqrt{v+1}}{\sqrt{2}}\Phi_{v+1}^{(0)}(\hat{z}) + \frac{\sqrt{v}}{\sqrt{2}}\Phi_{v-1}^{(0)}(\hat{z}), \\
 \frac{d\Phi_v^{(0)}(\hat{z})}{d\hat{z}} &= -\frac{\sqrt{v+1}}{\sqrt{2}}\Phi_{v+1}^{(0)}(\hat{z}) + \frac{\sqrt{v}}{\sqrt{2}}\Phi_{v-1}^{(0)}(\hat{z}),
 \end{aligned}
 \tag{21}$$

Table 2. The same as in Table I, but for $Q_{ij}(r)$ at $i \neq j$

i, j	$Q_{12}, 10^{-2}$	$Q_{23}, 10^{-2}$	$Q_{34}, 10^{-1}$	$Q_{13}, 10^{-2}$	$Q_{24}, 10^{-1}$	$Q_{14}, 10^{-4}$
$r^{-1}Q_{ij}^{(1)}$	6.1237243	8.6602540	1.0606601	-8.8388347	-1.5309310	0
$+r^{-3}Q_{ij}^{(3)}$	6.2072876	8.8911941	1.1027551	-8.8450495	-1.5340009	-5.5338541
$+r^{-5}Q_{ij}^{(5)}$	6.2085282	8.8962122	1.1040117	-8.8447852	-1.5339440	-5.6676737
$+r^{-7}Q_{ij}^{(7)}$	6.2085518	8.8963403	1.1040530	-8.8447748	-1.5339400	-5.6710585
$+r^{-9}Q_{ij}^{(9)}$	6.2085523	8.8963441	1.1040545	-8.8447745	-1.5339398	-5.6711548
(n.v.)	6.2085523	8.8963442	1.1040546	-8.8447745	-1.5339398	-5.6711580

$$\hat{z} \frac{d\Phi_v^{(0)}(\hat{z})}{d\hat{z}} = -\frac{1}{2}\Phi_v^{(0)}(\hat{z}) - \frac{\sqrt{v+1}\sqrt{v+2}}{2}\Phi_{v+2}^{(0)}(\hat{z}) + \frac{\sqrt{v-1}\sqrt{v}}{2}\Phi_{v-2}^{(0)}(\hat{z}),$$

$$L(n)\Phi_{n+s}^{(0)}(\hat{z}) \equiv \left(-\frac{d^2}{d\hat{z}^2} - (2n+1) + \hat{z}^2\right)\Phi_{n+s}^{(0)}(\hat{z}) = 2s\Phi_{n+s}^{(0)}(\hat{z}).$$

From (13), (15), and (21) we obtain the needed value of $M(k) = 2k + 1$ in the expansion (19) to provide the calculation of nonzero terms only.

Substituting Eq. (19) into Eq. (13), using Eq. (21) and equating coefficients at the same powers of r , we arrive at a set of recurrence relations for evaluating the coefficients $\beta_n^{(2k)}$ and $b_{n;s}^{(2k)}$

$$2sb_{n;s}^{(2k)} = f_{n;s}^{(2k)}, \tag{22}$$

$$f_{n;s}^{(2k)} = -\sum_{t=-4}^4 h_{n;s-t,t} b_{n;s-t}^{(2k-2)} - \sum_{j=1}^k \sum_{t=-2j}^{2j} a_{n;s-t,t}^{(2j)} b_{n;s-t}^{(2k-2j)} + \sum_{j=1}^k \beta_n^{(2j)} b_{n;s}^{(2k-2j)},$$

$$I_{jj'}^{(2k)} = \sum_{l=0}^k \sum_{s=-2k-1}^{2k+1} b_{n_l;s}^{(2l)} b_{n_r;s+n_l-n_r}^{(2k-2l)} = \delta_{k0} \delta_{n_l n_r}, \tag{23}$$

with the initial data $\beta_n^{(0)} = 2n + 1$ and $b_{n;s}^{(0)} = \delta_{s0}$. The coefficients $h_{n;s,t}$ and $a_{n;s,t}^{(2j)}$ in the relations (22) are calculated using (15), (21) from the relations

$$\frac{(\hat{z} - z_c\sqrt{\omega})^2}{\omega} \frac{d^2\Phi_{n+s}^{(0)}(\hat{z})}{d\hat{z}^2} + \frac{2(\hat{z} - z_c\sqrt{\omega})}{\omega} \frac{d\Phi_{n+s}^{(0)}(\hat{z})}{d\hat{z}} = \sum_{t=-4}^4 h_{n;s,t} \Phi_{n+s+t}^{(0)}(\hat{z}),$$

$$a^{(2j)}(\hat{z}, z_c) \Phi_{n+s}^{(0)}(\hat{z}) = \sum_{t=-2j}^{2j} a_{n;s,t}^{(2j)} \Phi_{n+s+t}^{(0)}(\hat{z}). \tag{24}$$

The corresponding coefficients $Q_{jj'}^{(2k+1)}$ and $H_{jj'}^{(2k+2)}$ from Eq. (18) have the following explicit form:

$$Q_{jj'}^{(2k+1)} = -\sum_{l=0}^k \sum_{s=-2k-1}^{2k+1} b_{n_l;s}^{(2l)} \left((-2k+2l) b_{n_r;s+n_l-n_r}^{(2k-2l)} \right) \tag{25}$$

Table 3. The same as in Table 1, but for $H_{ij}(r)$ at $i \neq j$

i, j	$H_{12}, 10^{-3}$	$H_{23}, 10^{-2}$	$H_{34}, 10^{-2}$	$H_{13}, 10^{-3}$	$H_{24}, 10^{-2}$	$H_{14}, 10^{-2}$
$r^{-2} H_{ij}^{(2)}$	-7.6546554	-2.1650635	-3.9774756	-5.3033008	-0.9185586	1.8750000
$+r^{-4} H_{ij}^{(4)}$	-7.7794422	-2.2165683	-4.1102192	-5.6189439	-1.0089114	1.9299804
$+r^{-6} H_{ij}^{(6)}$	-7.7817944	-2.2178064	-4.1142837	-5.6289351	-1.0129576	1.9313221
$+r^{-8} H_{ij}^{(8)}$	-7.7818496	-2.2178410	-4.1144203	-5.6292488	-1.0131247	1.9313584
$+r^{-10} H_{ij}^{(10)}$	-7.7818511	-2.2178422	-4.1144255	-5.6292592	-1.0131316	1.9313595
(n.v.)	-7.7818512	-2.2178422	-4.1144257	-5.6292596	-1.0131319	1.9313596

$$\begin{aligned}
 & + \frac{z_c \sqrt{\omega} \sqrt{n_l + s}}{\sqrt{2}} b_{n_r; s+n_l-n_r-1}^{(2k-2l)} - \frac{z_c \sqrt{\omega} \sqrt{n_l + s + 1}}{\sqrt{2}} b_{n_r; s+n_l-n_r+1}^{(2k-2l)} \\
 & - \frac{\sqrt{n_l + s - 1} \sqrt{n_l + s}}{2} b_{n_r; s+n_l-n_r-2}^{(2k-2l)} + \frac{\sqrt{n_l + s + 1} \sqrt{n_l + s + 2}}{2} b_{n_r; s+n_l-n_r+2}^{(2k-2l)} \Big), \\
 H_{jj'}^{(2k+2)} = & \sum_{l=0}^k \sum_{s=-2k-1}^{2k+1} b_{n_l; s}^{(2l)} \left(\left\{ 2l(2k-2l) + \frac{(n_l + s)^2 + n_l + s + 1}{2} \right. \right. \\
 & + \left. \frac{z_c^2 \omega}{2} (2n_l + 2s + 1) \right\} b_{n_r; s+n_l-n_r}^{(2k-2l)} \\
 & + \frac{z_c \sqrt{\omega} \sqrt{n_l + s}}{\sqrt{2}} (-4l + 2k - n_l - s) b_{n_r; s+n_l-n_r-1}^{(2k-2l)} \\
 & + \frac{z_c \sqrt{\omega} \sqrt{n_l + s + 1}}{\sqrt{2}} (4l - 2k - n_l - s - 1) b_{n_r; s+n_l-n_r+1}^{(2k-2l)} \\
 & - \frac{\sqrt{n_l + s - 1} \sqrt{n_l + s}}{2} (-4l + 2k + z_c^2 \omega) b_{n_r; s+n_l-n_r-2}^{(2k-2l)} \\
 & - \frac{\sqrt{n_l + s + 1} \sqrt{n_l + s + 2}}{2} (4l - 2k + z_c^2 \omega) b_{n_r; s+n_l-n_r+2}^{(2k-2l)} \\
 & + \frac{z_c \sqrt{\omega} \sqrt{n_l + s - 2} \sqrt{n_l + s - 1} \sqrt{n_l + s}}{\sqrt{2}} b_{n_r; s+n_l-n_r-3}^{(2k-2l)} \\
 & + \frac{z_c \sqrt{\omega} \sqrt{n_l + s + 1} \sqrt{n_l + s + 2} \sqrt{n_l + s + 3}}{\sqrt{2}} b_{n_r; s+n_l-n_r+3}^{(2k-2l)} \\
 & - \frac{\sqrt{n_l + s - 3} \sqrt{n_l + s - 2} \sqrt{n_l + s - 1} \sqrt{n_l + s}}{4} b_{n_r; s+n_l-n_r-4}^{(2k-2l)} \\
 & - \left. \frac{\sqrt{n_l + s + 1} \sqrt{n_l + s + 2} \sqrt{n_l + s + 3} \sqrt{n_l + s + 4}}{4} b_{n_r; s+n_l-n_r+4}^{(2k-2l)} \right). \tag{26}
 \end{aligned}$$

In **step 5** we sequentially evaluate the solutions $b_{n; s}^{(2k)}$ and $\beta_n^{(2k)}$ of the set of recurrence relations (22), (23) in each k th order ($k = 1, \dots, k_{\max}$):

$$\begin{aligned}
 f_{n; 0}^{(2k)} &= 0 \rightarrow \beta_n^{(2k)}; \\
 b_{n; s \neq 0}^{(2k)} &= f_{n; s}^{(2k)} / (2s); \\
 I_{ii}^{(2k)} &= \delta_{k0} \rightarrow b_{n; 0}^{(2k)}.
 \end{aligned}$$

Table 4. The same as in Table **1** but for $H_{jj}(r)$

j, j	$H_{11}, 10^{-2}$	$H_{22}, 10^{-2}$	$H_{33}, 10^{-2}$	$H_{44}, 10^{-1}$
$r^{-2}H_{jj}^{(2)}$	1.1562500	3.4687500	7.3437500	1.2781250
$+r^{-4}H_{jj}^{(4)}$	1.1675830	3.5283837	7.5051513	1.3110092
$+r^{-6}H_{jj}^{(6)}$	1.1677930	3.5299856	7.5110492	1.3125245
$+r^{-8}H_{jj}^{(8)}$	1.1677976	3.5300324	7.5112738	1.3125967
$+r^{-10}H_{jj}^{(10)}$	1.1677977	3.5300339	7.5112827	1.3126002
(n.v.)	1.1677977	3.5300339	7.5112831	1.3126004

In **step 6**, by substituting **(12)** with the coefficients $b_{n;s}^{(2k)}$ calculated at **step 5** into the expressions for the matrix elements evaluated at **step 4** and taking into account the above definition $r^{-2}E_j(r) = r^{-2}(\lambda_j(r) + c^2 + f) = \omega\beta_j(r)$, i.e. $E_j^{(2k)} = \omega\beta_j^{(2k)}$, we produce the **output** containing the matrix elements as an expansion in inverse powers of r for $k = 0, 1, \dots, k_{\max}$ at $j, j' = 1, \dots, j_{\max}$:

$$r^{-2}E_j(r) = \sum_{k=0}^{k_{\max}} \frac{E_j^{(2k)}}{r^{2k}}, \quad H_{jj'}(r) = \sum_{k=1}^{k_{\max}} \frac{H_{jj'}^{(2k)}}{r^{2k}}, \quad Q_{jj'}(r) = \sum_{k=1}^{k_{\max}} \frac{Q_{jj'}^{(2k-1)}}{r^{2k-1}}. \quad (27)$$

The calculation described above was performed by the algorithm implemented in MAPLE up to $k_{\max} = 8$. For example, the explicit expression of the desirable nonzero coefficients $E_j^{(2k)}, H_{ij}^{(2k)} = H_{ji}^{(2k)}$ and $Q_{ij}^{(2k-1)} = -Q_{ji}^{(2k-1)}$ reads as ($j = n + 1$):

$$\begin{aligned} E_j^{(0)} &= \omega(2n + 1), \quad E_j^{(2)} = m^2 - \frac{1}{4} - \frac{z_c^2\omega}{2}(2n + 1) - \frac{n^2 + n + 1}{2}, \\ H_{jj}^{(2)} &= \frac{z_c^2\omega}{2}(2n + 1) + \frac{n^2 + n + 1}{2}, \\ H_{jj-1}^{(2)} &= -\frac{z_c\sqrt{\omega}\sqrt{nn}}{\sqrt{2}}, \quad H_{jj-2}^{(2)} = -\frac{\omega z_c^2\sqrt{n-1}\sqrt{n}}{2}, \\ Q_{jj-1}^{(1)} &= -\frac{z_c\sqrt{\omega}\sqrt{n}}{\sqrt{2}}, \quad Q_{jj-2}^{(1)} = \frac{\sqrt{n-1}\sqrt{n}}{2}. \end{aligned} \quad (28)$$

Tables **1-4** demonstrate the convergence of partial sums in the asymptotic expansions **(27)** of effective potentials $Q_{ij}(r)$ and $H_{ij}(r)$ calculated by the algorithm MATRA to the corresponding numerical values calculated by means of ODPEVP **[9]**.

Remark 3. As follows from Eq. **(28)**, the reduction of the problem **(1)** under the axial symmetry at fixed m by means of the modified angular prolate spheroidal functions **(4)** at large r leads to the asymptotic centrifugal term $(E_j^{(2)} + H_{jj}^{(2)})r^{-2} = (m^2 - 1/4)r^{-2}$ in the effective potentials **(9)** of the set of radial equations **(8)**. The latter term is characterized by the integer magnetic quantum number m .

4 Symbolic Algorithm for Evaluation the Asymptotic Forms of Radial Solutions

In the procedure ASYMRS, using the above asymptotic expressions of the matrix elements, the asymptotic forms of the fundamental radial solutions $\chi_{ji_o}(r)$ of Eqs. (8) at small and large values of r are calculated, and the needed boundary conditions for the reduced interval $[r_{\min}, r_{\max}]$ are generated. Here we consider the case of large r .

We find the asymptotic solution $\chi_{ji_o}^{as}(r)$ at large $r \geq r_{\max}$ in the form

$$\chi_{ji_o}^{as}(r) = \left(\phi_{ji_o}(r) + \psi_{ji_o}(r) \frac{d}{dr} \right) R(p_{i_o}, r), \tag{29}$$

where $p_{i_o}^2 = 2E - \varepsilon_{i_o}^{th}$ is the relative energy with respect to the threshold value $\varepsilon_{i_o}^{th} = E_{i_o}^{(0)}$, and the function $R(p_{i_o}, r) \equiv R(p_{i_o}, r)$ satisfies the differential equation

$$\frac{d^2 R(p_{i_o}, r)}{dr^2} + \frac{2}{r} \frac{dR(p_{i_o}, r)}{dr} + \left(p_{i_o}^2 + \frac{2q}{r} - \frac{M_2}{r^2} \right) R(p_{i_o}, r) = 0 \tag{30}$$

In (30) the asymptotic centrifugal term M_2/r^2 with the factor $M_2 = m^2 - 1/4$ determines the order ν of the desirable solution

$$R(p_{i_o}, r) \equiv R_\nu(p_{i_o}, r) = p_{i_o}^{-1/2} r^{-1} ({}_1F_\nu(p_{i_o}, r) + G_\nu(p_{i_o}, r))/2,$$

where $F_\nu(p_{i_o}, r)$ and $G_\nu(p_{i_o}, r)$ are the regular and irregular Coulomb functions of the half-integer order $\nu = m - 1/2$ [11].

Remark 4. In the conventional 3D problem under spherical symmetry using the angular spherical harmonic functions leads to integer $M_2 = l(l + 1)$ and $\nu = l$ [12], whereas in the 3D problem under axial symmetry the angular oblate spheroidal functions at large r lead to $M_2 = 0$ and $\nu = 0$ [6]. However, at small r in both cases we have $M_2 = l(l + 1)$ and $\nu = l$.

In the case of $p_{i_o} = 0$ and $q \neq 0$ the function $R(p_{i_o}, r)$ has the form

$$R(p_{i_o}, r) = \pi^{1/2} r^{-1/2} ({}_1J_{\nu'}(\sqrt{8qr}) - Y_{\nu'}(\sqrt{8qr}))/2,$$

while in the case of $p_{i_o} \neq 0$ and $q = 0$ it reads as

$$R(p_{i_o}, r) = \pi^{1/2} 2^{-1/2} r^{-1/2} ({}_1J_{\nu'/2}(p_{i_o} r) - Y_{\nu'/2}(p_{i_o} r))/2.$$

Here $J_{\nu'}$ and $Y_{\nu'}$ are Bessel functions of the first and the second kind [10] of the order $\nu' = \sqrt{1 + 4M_2}$, $\nu' = 2m$ at $M_2 = m^2 - 1/4$ and $\nu' = 2l + 1$ at $M_2 = l(l + 1)$.

In **step 1** substituting the function (29) into Eq. (8), using (30) and extracting the coefficients for the Coulomb function and its derivative, we arrive at two coupled differential equations with respect to the unknown functions $\phi_{ji_o}(r)$ and $\psi_{ji_o}(r)$.

In **step 2** we expand the functions $\phi_{j i_o}(r)$ and $\psi_{j i_o}(r)$ in inverse powers of r :

$$\phi_{j i_o}(r) = \sum_{k=0}^{k_{\max}} \phi_{j i_o}^{(k)} r^{-k}, \quad \psi_{j i_o}(r) = \sum_{k=0}^{k_{\max}} \psi_{j i_o}^{(k)} r^{-k}. \quad (31)$$

After substituting the expansions (27), (31) into Eqs. (8) and equating the coefficients at the same powers of r , we compute a set of recurrence relations with respect to the unknown coefficients $\phi_{j i_o}^{(k)}$ and $\psi_{j i_o}^{(k)}$

$$\left(p_{i_o}^2 - 2E + E_j^{(0)}\right) \phi_{j i_o}^{(k)} = f_{j i_o}^{(k)}, \quad \left(p_{i_o}^2 - 2E + E_j^{(0)}\right) \psi_{j i_o}^{(k)} = g_{j i_o}^{(k)}, \quad (32)$$

where the right-hand sides $f_{j i_o}^{(k)}$ and $g_{j i_o}^{(k)}$ are defined by the relations

$$f_{j i_o}^{(k)} = 2p_{i_o}^2(k-1)\psi_{j i_o}^{(k-1)} \quad (33)$$

$$+ \left((k-2)(k-3) + M_2 - \left(E_j^{(2)} + H_{jj}^{(2)}\right)\right) \phi_{j i_o}^{(k-2)} + 2q(2k-3)\psi_{j i_o}^{(k-2)}$$

$$- M_2(k-2)\psi_{j i_o}^{(k-3)} - \sum_{k'=3}^k \left(E_j^{(k')} + H_{jj}^{(k')}\right) \phi_{j i_o}^{(k-k')}$$

$$+ \sum_{j'=1, j' \neq j}^{j_{\max}} \sum_{k'=1}^k \left[\left((2k-k'-3)Q_{jj'}^{(k'-1)} - H_{jj'}^{(k')}\right) \phi_{j' i_o}^{(k-k')}\right.$$

$$\left. + \left(2p_{i_o}^2 Q_{jj'}^{(k')} + 4qQ_{jj'}^{(k'-1)}\right) \psi_{j' i_o}^{(k-k')}\right],$$

$$g_{j i_o}^{(k)} = -2(k-1)\phi_{j i_o}^{(k-1)} \quad (34)$$

$$+ M_2\phi_{j i_o}^{(k-2)} + \left(k(k-1) - \left(E_j^{(2)} + H_{jj}^{(2)}\right)\right) \psi_{j i_o}^{(k-2)}$$

$$- 2M_2(k-2)\psi_{j i_o}^{(k-3)} - \sum_{k'=1}^k \left(E_j^{(k')} + H_{jj}^{(k')}\right) \psi_{j i_o}^{(k-k')}$$

$$+ \sum_{j'=1, j' \neq j}^{j_{\max}} \sum_{k'=1}^k \left[\left((2k-k'+1)Q_{jj'}^{(k'-1)} - H_{jj'}^{(k')}\right) \psi_{j' i_o}^{(k-k')} - 2Q_{jj'}^{(k')} \phi_{j' i_o}^{(k-k')}\right].$$

It should be noted that these relations differ from the case of $M_2 = 0$ [6] only by the terms containing $M_2 = E_{i_o}^{(2)} + H_{i_o i_o}^{(2)}$.

In **step 3** from equations (32) at $k = 0$ we get the initial data for the recurrence procedure, including the special threshold case $2E = E_{i_o}^{(0)}$ ($p_{i_o}^2 = 0$)

$$\phi_{j_0 i_o}^{(0)} = \delta_{j_0 i_o}, \quad \psi_{j_0 i_o}^{(0)} = 0, \quad p_{i_o}^2 = 2E - E_{i_o}^{(0)}. \quad (35)$$

The open channels have $p_{i_o}^2 \geq 0$ whereas the close channels have $p_{i_o}^2 < 0$. Suppose that there are $N_o \leq j_{\max}$ open channels, i.e., $p_{i_o}^2 \geq 0$ for $i_o = 1, \dots, N_o$ and $p_{i_o}^2 < 0$ for $i_o = N_o + 1, \dots, j_{\max}$. After substitution of (35) into (32) the recurrence relations for $k = 1, 2, \dots, k_{\max}$ take the form

$$\left(E_j^{(0)} - E_{i_o}^{(0)}\right) \phi_{j i_o}^{(k)} = f_{j i_o}^{(k)}, \quad \left(E_j^{(0)} - E_{i_o}^{(0)}\right) \psi_{j i_o}^{(k)} = g_{j i_o}^{(k)}. \quad (36)$$

Step 4 performs the calculation of the coefficients $\phi_{j i_o}^{(k)}$ and $\psi_{j i_o}^{(k)}$ by a step-by-step procedure of solving equations (36) with r.h.s. determined by (33), (34), for $k = 1, 2, \dots, k_{\max}$:

$$\begin{aligned} \phi_{j i_o}^{(k)} &= \frac{f_{j i_o}^{(k)}}{E_j^{(0)} - E_{i_o}^{(0)}}, & \psi_{j i_o}^{(k)} &= \frac{g_{j i_o}^{(k)}}{E_j^{(0)} - E_{i_o}^{(0)}}, & j &\neq i_o, \\ \{f_{i_o i_o}^{(k+1+\delta p_{i_o})} = 0, g_{i_o i_o}^{(k+1)} = 0\} &\rightarrow \{\phi_{i_o i_o}^{(k)}, \psi_{i_o i_o}^{(k)}\}, \end{aligned}$$

where $\delta p_{i_o} = 1$ at $p_{i_o} = 0$ (threshold case) and $\delta p_{i_o} = 0$ at $p_{i_o} \neq 0$.

The calculation was performed by means of the algorithm, implemented in MAPLE up to $k_{\max} = 8$. Its **output** contains the elements at $k = 0, 1, \dots, k_{\max}$. For example, for $k_{\max} = 1$ we have the coefficients $\phi_{j i_o}^{(k)}$ and $\psi_{j i_o}^{(k)}$ in the form

$$\begin{aligned} \phi_{j i_o}^{(1)} &= 0, & \psi_{j i_o}^{(1)} &= \frac{2Q_{j i_o}^{(1)}}{E_j^{(0)} - E_{i_o}^{(0)}}, & (37) \\ \phi_{i_o i_o}^{(1)} &= 0, & \psi_{i_o i_o}^{(1)} &= -\left(1 - \frac{\delta p_{i_o}}{3}\right) \sum_{j_0=\max(1, i_o-2), j_0 \neq i_o}^{\min(j_{\max}, i_o+2)} Q_{i_o j_0}^{(1)} \psi_{j_0 i_o}^{(1)}, \end{aligned}$$

and substituting the asymptotic expressions (27) into the above equation, we arrive at the explicit expression of the desirable nonzero coefficients $\phi_{j i_o}^{(k)}$ and $\psi_{j i_o}^{(k)}$ (for $j_{\max} \geq i_o + 2k$, $i_o = n_o + 1$, $\langle \hat{z} | i_o \rangle = \sqrt[4]{\omega} \Phi_{n_o}^{(0)}(\hat{z})$, $z' = z - z_c = \frac{\hat{z}}{\sqrt{\omega}} - z_c$):

$$\begin{aligned} \psi_{i_o-2i_o}^{(1)} &= -\frac{1}{2} \frac{\sqrt{n_o-1} \sqrt{n_o}}{2\omega} = -\frac{1}{2} \left\langle i_o - 2 \left| (z - z_c)^2 \right| i_o \right\rangle_z, \\ \psi_{i_o-1i_o}^{(1)} &= \frac{z_c \sqrt{n_o}}{\sqrt{2} \sqrt{\omega}} = -\frac{1}{2} \left\langle i_o - 1 \left| (z - z_c)^2 \right| i_o \right\rangle_z, & (38) \\ \psi_{i_o i_o}^{(1)} &= -\frac{1}{2} \left(1 - \frac{\delta p_{i_o}}{3}\right) \left(\frac{2n_o + 1}{2\omega} + z_c^2\right) = -\frac{1}{2} \left(1 - \frac{\delta p_{i_o}}{3}\right) \left\langle i_o \left| (z - z_c)^2 \right| i_o \right\rangle_z, \\ \psi_{i_o+1i_o}^{(1)} &= \frac{z_c \sqrt{n_o+1}}{\sqrt{2} \sqrt{\omega}} = -\frac{1}{2} \left\langle i_o + 1 \left| (z - z_c)^2 \right| i_o \right\rangle_z, \\ \psi_{i_o+2i_o}^{(1)} &= -\frac{1}{2} \frac{\sqrt{n_o+1} \sqrt{n_o+2}}{2\omega} = -\frac{1}{2} \left\langle i_o + 2 \left| (z - z_c)^2 \right| i_o \right\rangle_z. \end{aligned}$$

Remark 5. The obtained results correspond to the asymptotic transformation of the arguments r, η of the total function $\psi_{m i_o}^{as}$ in terms of the asymptotic basis functions $\Phi_j^{as}(\hat{z}; r)$ from (12) at fixed magnetic quantum number m :

$$\begin{aligned} \psi_{m i_o}^{as}(r, \eta) &= \sum_{j=1}^{j_{\max}} \Phi_j^{as}(\hat{z}; r) \chi_{j i_o}^{as}(r) = \sum_{j=1}^{j_{\max}} \Phi_j^{as}(\hat{z}; r) \left(\phi_{j i_o}(r) + \psi_{j i_o}(r) \frac{d}{dr} \right) R(p_{i_o}, r) \\ &= \sqrt[4]{\omega} \sqrt{r} \sum_{k=0}^{k_{\max}} r^{-k} \sum_{p=0}^k \sum_{j=1}^{j_{\max}} \Phi_j^{(p)}(\hat{z}) \left(\phi_{j i_o}^{(k-p)} + \psi_{j i_o}^{(k-p)} \frac{d}{dr} \right) R(p_{i_o}, r). \end{aligned}$$

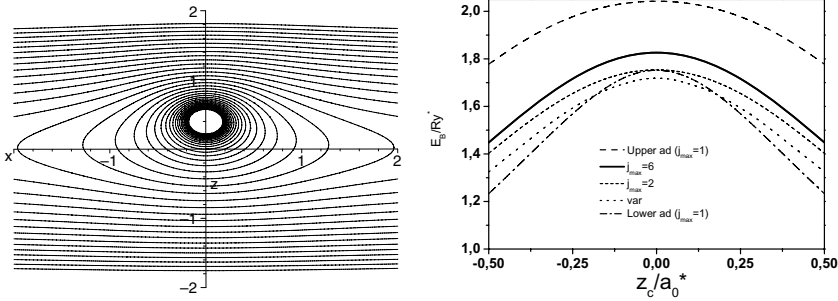


Fig. 1. Isolines of the potential energy surface as a function of two independent variables ρ, z with shift of center of Coulomb potential along the variable z on $z_c = 0.4$ (left panel). The binding energy $-E_B = 2E - \varepsilon_1^h$ ($\text{Ry}^* = 5.2 \text{ meV}$) versus position shift of Coulomb center of impurity by variable z in interval $z_c \in [-0.5, 0.5]$ ($a_0^* = 102 \text{ \AA}$) at $q = 1, \omega = 3$ and $m = 0$ (right panel). Dotted line is variational calculations [1], dash-dotted line is the adiabatic approximation ($j_{max} = 1$), short-dashed line is the Kantorovich approximation [4] at $j_{max} = 2$ basis functions, solid line is the Kantorovich approximation [4] at $j_{max} = 6$ basis functions, dashed line is the crude adiabatic approximation ($j_{max} = 1$, when the diagonal adiabatic positive correction $H_{11}(r) = 0$ is neglected).

Taking into account the orthogonality [14] and completeness $\sum_j \langle \hat{z}' | j \rangle \langle j | \hat{z} \rangle = \delta(\hat{z}' - \hat{z})$ relations for the asymptotic basis functions $\Phi_j^{(0)}(\hat{z}; r)$, we obtain the desirable asymptotic form of the total wave function at $p_{i_o} \hat{z} / (2r) \ll 1$ and $k_{max} = 1$:

$$\begin{aligned}
 \psi_{m_{i_o}}^{(as)}(r, \eta) &= \sqrt[4]{\omega} \sqrt{r} \sum_j \langle \hat{z} | j \rangle \left[\langle j | i_o \rangle - \frac{1}{2r} \langle j | (z - z_c)^2 | i_o \rangle \frac{d}{dr} \right] R(p_{i_o}, r) \\
 &\approx \sqrt[4]{\omega} \sqrt{r} \Phi_{i_o}^{(0)}(\hat{z}) \chi_{i_o i_o}^{(as)} \left(r - \frac{(z - z_c)^2}{2r} \right) \\
 &\approx \frac{\sqrt[4]{\omega}}{2(p_{i_o} \rho)^{1/2}} \Phi_{i_o}^{(0)}(\hat{z}) (\nu F_\nu(p_{i_o}, \rho) + G_\nu(p_{i_o}, \rho)).
 \end{aligned} \tag{39}$$

Thus, we obtain the needed compatibility conditions for the asymptotic solutions of scattering problems in the spherical coordinates (r, η, φ) shifted by z_c along z axis and in the cylindrical coordinates (ρ, z, φ)

$$\rho = r \sqrt{1 - (z - z_c)^2 / r^2} = r - (z - z_c)^2 / (2r) + O(r^{-2}),$$

including the regular F_ν and irregular G_ν Coulomb functions of the half-integer order $\nu = m - 1/2$ from Eq. (30).

It should be noted that at large r the linearly independent functions (29) satisfy the Wronskian-type relation

$$\mathbf{W}(\mathbf{Q}(r); \chi^*(r), \chi(r)) = \frac{2}{\rho} \mathbf{I}_{o_o}, \tag{40}$$

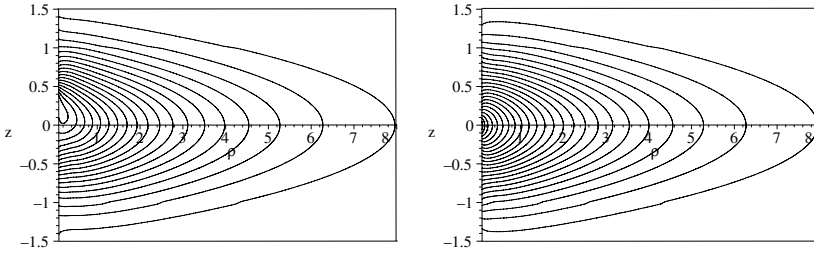


Fig. 2. Isolines of the ground-state wave function for the values of parameters $q = 1$, $\omega = 3$ and $m = 0$. Left panel: $z_c = 0.4$, Right panel: $z_c = 0$.

where $\mathbf{W}(\bullet; \chi^*(r), \chi(r))$ is the generalized Wronskian with the long derivative defined as

$$\mathbf{W}(\bullet; \chi^*(r), \chi(r)) = r^2 \left[(\chi^*)^T \left(\frac{d\chi}{dr} - \bullet \chi \right) - \left(\frac{d\chi^*}{dr} - \bullet \chi^* \right)^T \chi \right].$$

These relations will be used to examine the desirable accuracy of the above expansion till k_{\max} using KANTBP program implemented in FORTRAN [5,7] for numerical solving the boundary problem of discrete or continuous spectrum. The symbolic calculations of the above asymptotic expressions were performed using the codes MATRA and ASYMRS implemented in MAPLE, that generate FORTRAN codes of subroutines POTCAL and ASYMSC in KANTBP [5,7].

5 Test Example

The calculation for the GaAs parabolic quantum well was carried out with the values of parameters $q = 1$, $\omega = 3$, $m = 0$, and $z_c \in [-0.5, 0.5]$ in the reduced atomic units from [14] by applying the programs KANTBP [5] and ODPEVP [9]. These programs implementing the finite-element method to solve the boundary-value problems (8)–(11) and (5)–(7) were applied respectively on the grids $\Omega_r = \{0(200)1(200)5(200)100\}$ and $\Omega_\eta = \{-1(800)1\}$ with the Lagrange elements of the order $p = 4$ between the nodes. In the above grids Ω_r and Ω_η , the number of grid elements is shown in the parentheses.

As follows from the theorem [13], for the ground state the adiabatic approximation ($j_{\max} = 1$) gives the upper bound for the energy, while in the so-called crude adiabatic approximation, when the diagonal adiabatic positive correction $H_{11}(r) = 0$ is neglected, one gets the lower bound for the energy. The corresponding inverse estimators for the binding energy $-E_B = 2E - \varepsilon_1^{th}$ (in units $Ry^* = 5.2$ meV) in spherical coordinates are presented in Fig. 1. As one can see, these values are upper and lower estimates of the binding energy from the variational calculation [1]. The corresponding inverse lower estimators of the binding energy for increasing number of single-parameter basis functions j_{\max}

allow one to analyze the convergence rate of the method used for solving the boundary-value problem in the two-dimensional domain (see Fig. 2). In particular, the Kantorovich approximation (4) at $j_{max} = 10$ basis functions leads to the following inverse lower estimation of the binding energy $E_B/Ry^* = 1.82774$.

6 Conclusion

We presented the scheme for solving the boundary-value problem with discrete spectrum for a parabolic quantum well in the adiabatic representation. The upper and lower bounds for the energy of the ground state of the systems are obtained under the conditions of the shift of the Coulomb center in a given range of the parameter with respect to earlier variational estimates. It is shown that the rate of convergence depends significantly on the appropriate choice of the adiabatic basis parameterization taking the specific features of the considered problem into account. The presented results allow one to estimate the efficiency of the method and to prove the compatibility conditions (39) for asymptotic solutions of scattering problems in spherical and cylindrical coordinates. The software package developed is applicable to the investigation of semiconductor nanostructure models. Further development of the method and the software package is planned for solving the quasi-2D and quasi-1D boundary-value problems with both discrete and continuous spectrum, which are necessary for calculating the optical transition rates, channelling and transport characteristics in the models like quantum wells and quantum wires.

Acknowledgements. The authors thank Profs. V.L. Derbov, E.M. Kazaryan, A.A. Kostanyan and H.A. Sarkisyan for collaboration. The work was supported partially by RFBR (grants 07-01-00660 and 08-01-00604).

References

1. Kazaryan, E.M., Kostanyan, A.A., Sarkisyan, H.: Impurity optical absorption in parabolic quantum well. *Physica E* 28, 423–430 (2005)
2. Voss, H.: Numerical calculation of the electronic structure for three-dimensional quantum dots. *Comput. Phys. Commun.* 174, 441–446 (2006)
3. Wang, W., Hwang, T.-M., Jang, J.-C.: A second-order finite volume scheme for three dimensional truncated pyramidal quantum dot. *Comput. Phys. Commun.* 174, 371–385 (2006)
4. Gusev, A.A., Chuluunbaatar, O., Vinitsky, S.I., Derbov, V.L., Kazaryan, E.M., Kostanyan, A.A., Sarkisyan, H.A.: Adiabatic approach to the problem of a quantum well with a hydrogen-like impurity. *Phys. Atomic Nuclei* 72, 1600–1608 (2010)
5. Chuluunbaatar, O., Gusev, A.A., Abrashkevich, A.G., Amaya-Tapia, A., Kaschiev, M.S., Larsen, S.Y., Vinitsky, S.I.: KANTBP: A program for computing energy levels, reaction matrix and radial wave functions in the coupled-channel hyperspherical adiabatic approach. *Comput. Phys. Commun.* 177, 649–675 (2007)

6. Chuluunbaatar, O., Gusev, A.A., Gerdt, V.P., Rostovtsev, V.A., Vinitsky, S.I., Abrashkevich, A.G., Kaschiev, M.S., Serov, V.V.: POTHMF: A program for computing potential curves and matrix elements of the coupled adiabatic radial equations for a hydrogen-like atom in a homogeneous magnetic field. *Comput. Phys. Commun.* 178, 301–330 (2008)
7. Chuluunbaatar, O., Gusev, A.A., Vinitsky, S.I., Abrashkevich, A.G.: KANTBP 2.0: New version of a program for computing energy levels, reaction matrix and radial wave functions in the coupled-channel hyperspherical adiabatic approach. *Comput. Phys. Commun.* 179, 685–693 (2008)
8. Gusev, A.I., Rempel, A.A.: *Nanocrystalline materials*. Cambridge Int. Sci, Cambridge (2004)
9. Chuluunbaatar, O., Gusev, A.A., Vinitsky, S.I., Abrashkevich, A.G.: ODPEVP: A program for computing eigenvalues and eigenfunctions and their first derivatives with respect to the parameter of the parametric self-adjointed Sturm-Liouville problem. Accepted in *Comput. Phys. Commun.* (2009), 10.1016/j.cpc.2009.04.017
10. Abramowitz, M., Stegun, I.A.: *Handbook of Mathematical Functions*. Dover, New York (1972)
11. Barnett, A.R.: KLEIN: Coulomb functions for real λ and positive energy of high accuracy. *Comput. Phys. Commun.* 24, 141–159 (1981)
12. Barnett, A.R., Feng, D.H., Steed, J.W., Goldfarb, L.J.B.: Coulomb wave functions for all real η and ρ . *Comput. Phys. Comm.* 8, 377–395 (1974)
13. Epstein, S.T.: Ground state energy of a molecule in adiabatic approximation. *J. Chem. Phys.* 144, 836–837 (1965)

New Analytic Solutions of the Problem of Gas Flow in a Casing with Rotating Disc

Evgenii V. Vorozhtsov

Khristianovich Institute of Theoretical and Applied Mechanics, Russian Academy of Sciences, Novosibirsk 630090, Russia
vorozh@itam.nsc.ru

Abstract. We analyse the known approximate analytic solution of the problem of gas flow induced by the disc rotation inside a closed casing. It is shown that this solution is inapplicable because of the negative thickness of the boundary layer in the shaft neighborhood. Several new analytic solutions are obtained for the flow parameters inside the boundary layer of the casing motionless base. To reduce further the discrepancy between the analytic solution and the direct difference solution of three-dimensional Navier–Stokes equations it is proposed to account for the viscous friction force moment on the lateral casing wall. The consideration of this moment has improved considerably the accuracy of the approximate analytic solution.

1 Introduction

According to [16], 8 % of the entire produced electric power were consumed in the former Soviet Union by the ventilators (fans). In this connection, the problem of the development of highly efficient and economical ventilation devices for pumping the gases, cleaning the gases and liquids from admixtures, etc., in various branches of the industry and agriculture is topical.

One can identify the following types of fans: centrifugal fans, axial fans, and disc fans. The disc fans may be considered to belong to friction machines. They differ from the conventional centrifugal and axial fans, which are the machines of dynamic action.

N. Tesla [17] was the first to propose the application of the friction principle in fans. Figure 1 shows the simplest (model) disc fan. The upper lid of the casing has been removed to show more clearly the peculiarities of the disc fan design. The arrows pointing to the left show the direction of the motion of gas sucked in the fan, and the arrows pointing to the right show the direction of the motion of gas ejected from the fan (the discs rotate counter-clockwise). The model fan shown in Fig. 1 contains only two discs. In real industrial disc fans, the number of discs on the shaft may amount to several dozens.

The disc fans are applied both in the industry and in agriculture [14, 5, 12]. The disc machines possess the following merits [5]: simplicity of the design; high anti-cavitation characteristics; low noise; stability of operation, stable supply of the gas or liquid.

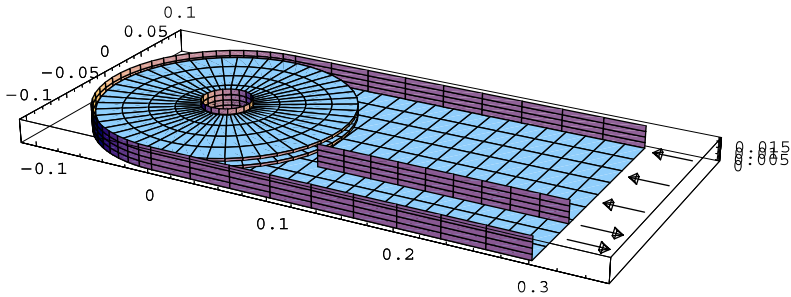


Fig. 1. The simplest disc fan

As follows from Fig. 1, the gas flow inside the disc fan cannot be axisymmetric because of the presence of channels with parallel walls. Therefore, the numerical modelling of the gas flow inside the fan under study must base on the mathematical model of a three-dimensional unsteady gas flow. In the existing disc fans the angular velocity of the rotation of discs is low, it ranges from 20 to 100 revolutions per minute (rpm). Therefore, the Mach number of the gas flow, as simple estimates show, does not exceed 0.05. In this connection, one can use the system of Navier–Stokes equations governing the viscous incompressible fluid flows. For the difference approximation of this system we have used a finite-difference scheme described in [18]. Due to the fact that there are solid walls, which do not coincide with the coordinate lines in the disc fan design, it is convenient to use the immersed boundary method described in [3] for computation of the gas flow in such fans.

For the verification of the developed computer code we have used the following two well-known analytical solutions. One of them is the solution of the problem of the Couette flow between two coaxial cylinders [7]. If we increase the thickness of each disc in a disc fan so that all the discs merge into a single rotating cylinder, we arrive at the known Couette problem. We present below in Section 3 the results of the comparison of the finite-difference solution by the method of [18,3] with the exact Couette–Taylor solution. These comparisons point to a correct work of the method of [18,3] implemented by us.

As the second analytical test for the developed computer code we have chosen an approximate analytic solution obtained by F. Schultz-Grunow [15] for the case of the rotation of a single disc in a closed cylindrical casing. The main results of the work [15] were reproduced subsequently without changes in the book [9] and in a more shortened form in the book [10].

The computer implementation of the approximate analytic solution [15] showed that the formula presented in [15,9,10] for the boundary layer thickness $\delta_0(x)$ on the motionless casing base yields the negative values of $\delta_0(x)$ near the rotor shaft, where x is a dimensionless coordinate. In this connection, we have repeated all the derivations of the work [15] with the aid of symbolic computations in the system *Mathematica*. Another example showing the usefulness of symbolic computations on computer for revealing the errors at the obtaining of analytic

solutions of theoretical mechanics problems was previously presented in [14]. We show below in Section 4 that the coefficients of the polynomial representation of the solution, which were presented in [15], do not satisfy the requirements formulated in [15]. In this connection, we propose in Section 5 to search for the coefficients entering the analytic solution by the method of least squares. As a result, a set of coefficients has been obtained, which ensures the non-negativeness of the boundary layer thickness on the motionless casing base.

The approximate analytic solution was presented in [15,9,10] in a form involving two polynomials $P_1(x)$ and $P_2(x)$ of the fourth degree in x . For the purpose of reducing the value of the objective function of the method of the least squares we present in Section 6 several new approximate analytic solutions, which we have obtained with the aid of the *Mathematica* program developed by us for several combinations of the degrees of polynomials P_1 and P_2 in the interval $4 \leq \text{Deg}(P_1), \text{Deg}(P_2) \leq 6$. These solutions are compared with one another as well as with the corrected solution obtained by us for the case $\text{Deg}(P_1) = \text{Deg}(P_2) = 4$.

2 Governing Equations and Finite-Difference Method

The Navier–Stokes equations governing unsteady three-dimensional laminar flows of a viscous incompressible fluid in the cylindrical coordinates θ, r, z , where θ is the azimuthal coordinate, r is the polar radius, and the z -axis is directed along the shaft on which the discs are mounted, may be written as [18]

$$\begin{aligned} \partial v_\theta / \partial t &= H_1 - (1/(\rho r)) \partial p / \partial \theta + (A_{1\theta} + A_{1r} + A_{1z}) v_\theta; \\ \partial q_r / \partial t &= H_2 - (r/\rho) \partial p / \partial r + (A_{2\theta} + A_{2r} + A_{2z}) q_r; \end{aligned} \tag{1}$$

$$\begin{aligned} \partial v_z / \partial t &= H_3 - (1/\rho) \partial p / \partial z + (A_{3\theta} + A_{3r} + A_{3z}) v_z; \\ \frac{\partial q_r}{\partial r} + \frac{\partial v_\theta}{\partial \theta} + r \frac{\partial v_z}{\partial z} &= 0, \end{aligned} \tag{2}$$

where

$$\begin{aligned} H_1 &= -\frac{1}{r^2} \frac{\partial r v_\theta q_r}{\partial r} - \frac{1}{r} \frac{\partial v_\theta^2}{\partial \theta} - \frac{\partial v_\theta v_z}{\partial z} + \frac{2\nu}{r^3} \frac{\partial q_r}{\partial \theta}; \\ H_2 &= -\frac{\partial}{\partial r} \left(\frac{q_r^2}{r} \right) - \frac{\partial}{\partial \theta} \left(\frac{v_\theta q_r}{r} \right) - \frac{\partial q_r v_z}{\partial z} + v_\theta^2 + \frac{2\nu}{r} \frac{\partial v_\theta}{\partial \theta}; \\ H_3 &= -\frac{1}{r} \frac{\partial q_r v_z}{\partial r} - \frac{\partial v_z^2}{\partial z}; \\ A_{1\theta} v_\theta &= \nu \left(-\frac{v_\theta}{r^2} + \frac{1}{r^2} \frac{\partial^2 v_\theta}{\partial \theta^2} \right); \quad A_{1r} v_\theta = \nu \left[\frac{1}{r} \left(\frac{\partial}{\partial r} r \frac{\partial v_\theta}{\partial r} \right) - \frac{v_\theta}{r^2} \right]; \\ A_{1z} v_\theta &= \nu \frac{\partial^2 v_\theta}{\partial z^2}; \\ A_{2\theta} q_r &= \frac{\nu}{r^2} \frac{\partial^2 q_r}{\partial \theta^2} - \frac{\partial}{\partial \theta} \left(\frac{v_\theta q_r}{r} \right); \quad A_{2r} q_r = \nu r \frac{\partial}{\partial r} \left(\frac{1}{r} \frac{\partial q_r}{\partial r} \right); \quad A_{2z} q_r = \nu \frac{\partial^2 q_r}{\partial z^2}; \\ A_{3\theta} v_z &= \frac{\nu}{r^2} \frac{\partial^2 v_z}{\partial \theta^2} - \frac{1}{r} \frac{\partial v_\theta v_z}{\partial \theta}; \quad A_{3r} v_z = \frac{\nu}{r} \frac{\partial}{\partial r} \left(r \frac{\partial v_z}{\partial r} \right); \quad A_{3z} v_z = \nu \frac{\partial^2 v_z}{\partial z^2}. \end{aligned}$$

In (1), (2), $v_\theta, v_r,$ and v_z are the components of the gas velocity vector along the $\theta-, r-,$ and $z-$ axes, respectively. Instead of $v_r,$ the dependent variable $q_r = r \cdot v_r$ is used to facilitate the treatment of the problem of the approximation of the Navier–Stokes equations at point $r = 0;$ p is the pressure, $\rho = \text{const} > 0$ is the density of the gas or liquid, $\nu = \mu/\rho, \mu = \text{const} > 0$ is the coefficient of the dynamic viscosity of gas.

The boundary conditions for system of equations (1), (2):

1°. The boundary conditions on the horizontal surfaces of the rotating disc:

$$v_r = 0, \quad v_\theta = \omega r, \quad v_z = 0, \quad \partial p/\partial z = 0,$$

where ω is the user-specified constant angular velocity of the disc rotation.

2°. The boundary conditions on the motionless butt ends of the cylindrical casing:

$$v_r = 0, \quad v_\theta = 0, \quad v_z = 0, \quad \partial p/\partial z = 0.$$

3°. The boundary conditions on the shaft and on the vertical butt end of the rotating disc:

$$v_r = 0, \quad v_\theta = \omega r, \quad v_z = 0, \quad \partial p/\partial r = \rho\omega^2 r.$$

Thus, the boundary conditions on solid walls for velocity components are the no-slip conditions.

The preliminary velocity field was computed by the one-stage difference scheme from [6]. We now briefly describe this scheme below. To ensure a short form of writing the scheme we introduce the notations: $q_1 \equiv v_\theta, q_2 \equiv q_r, q_3 \equiv v_z.$ Then the scheme from [6] as applied to system (1) has the form

$$(\hat{q}_i - q_i^n)/\tau_n = 0.5 (3H_i^n - H_i^{n-1}) - G_i p^n + (A_{i\theta h} + A_{irh} + A_{izh})(\hat{q}_i + q_i^n)/2, \quad (3)$$

where $i = 1, 2, 3, \tau_n$ is the time step, n is the time level number, $n = 0, 1, 2, \dots, A_{i\theta h}, A_{irh}, A_{izh}$ are the central difference approximations of the operators $A_{i\theta}, A_{ir}, A_{iz},$ respectively, on a nonuniform spatial grid; $G_i p^n$ is the difference approximation of the pressure term in the i th equation. A direct solution of the system of algebraic equations (3) is computationally very intensive, therefore, the approximate factorization scheme was used instead of (3) in [6,18,3]:

$$\begin{aligned} & \left(I - \frac{\tau_n}{2} A_{i\theta h}\right) \left(I - \frac{\tau_n}{2} A_{irh}\right) \left(I - \frac{\tau_n}{2} A_{izh}\right) \Delta \hat{q}_i \\ & = \tau_n [(3H_i^n - H_i^{n-1})/2 - G_i p^n + (A_{i\theta h} + A_{irh} + A_{izh})q_i^n], \end{aligned} \quad (4)$$

where I is the identity operator, $\Delta \hat{q}_i = \hat{q}_i - q_i^n.$ As was pointed out in [6], scheme (4) has the second order of approximation in the spatial variables and in time.

The intermediate velocity field does generally not satisfy the continuity equation. In order to ensure the conservation of mass the field of intermediate velocities is corrected at the second fractional step by the formula

$$(q_i^{n+1} - \hat{q}_i)/\tau_n = -G_i \varphi^{n+1}, \quad (5)$$

where the pressure correction φ^{n+1} is computed in the way to obtain a divergence-free velocity field at the $(n + 1)$ th time step. To this end let us apply the divergence operator to the both sides of equation (5):

$$(\nabla_h q_i^{n+1} - \nabla_h \hat{q}_i) / \tau_n = -\nabla_h G_i \varphi^{n+1}, \tag{6}$$

where the difference operator ∇_h is a difference approximation of the divergence operator on the left-hand side of equation (2): $\nabla q = \frac{\partial q_1}{\partial \theta} + \frac{\partial q_2}{\partial r} + r \frac{\partial q_3}{\partial z}$, and $L_h = \nabla_h G_i \varphi$ is the difference approximation of the elliptic operator $\nabla G_i \varphi = \frac{\partial}{\partial \theta} \left(\frac{1}{r} \frac{\partial \varphi}{\partial \theta} \right) + \frac{\partial}{\partial r} \left(r \frac{\partial \varphi}{\partial r} \right) + r \frac{\partial^2 \varphi}{\partial z^2}$. In view of the requirement $\nabla_h q_i^{n+1} = 0$, equation (6) is the Poisson equation for φ^{n+1} . Having found φ^{n+1} by solving equation (6), we compute the pressure p^{n+1} by formula [18,3] $p^{n+1} = p^n + \varphi^{n+1} - (\tau_n/2)\nu L_h \varphi^{n+1}$. The solution of the system of equations (4) was found with the aid of three tridiagonal inversions. The Poisson equation (6) was solved with the aid of the spectral/difference method described in [8]. Since the solution is 2π -periodic in the variable θ , the discrete Fourier transform in θ of the difference equation (6) was at first performed. The obtained system of linear algebraic equations for the complex coefficients of the discrete Fourier transform was then solved by the method of matrix sweep [13]. The direct and inverse discrete Fourier transforms were implemented with the aid of the algorithms of the discrete fast Fourier transform described in [11].

3 Couette Flow

Let us present the formulas of the exact solution for the Couette flow in the gap between two coaxial cylinders in the particular case when the internal cylinder rotates, and the external cylinder is at rest (see Fig. 2). We preliminarily perform the nondimensionalization of variables by formulas

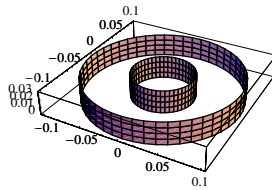


Fig. 2. Two coaxial cylinders in the Couette flow problem

$$r^* = \frac{r}{R_2}, \quad z^* = \frac{z}{R_2}, \quad v_\theta^* = \frac{v_\theta}{q_{ref}}, \quad v_r^* = \frac{v_r}{q_{ref}}, \quad v_z^* = \frac{v_z}{q_{ref}}, \quad p^* = \frac{p}{\rho q_{ref}^2}, \tag{7}$$

where the superscript * stands by the dimensionless variables, R_2 is the radius of the internal cylinder, $q_{ref} = \omega R_2$, ω is the angular velocity of the rotation of the internal cylinder. Denote by R_3 the dimensional radius of the external cylinder. The Couette flow is known to be stationary, that is the components

of the solution vector of the Navier–Stokes equations (11), (12) do not depend on time t [7]. Besides, the flow under consideration is axisymmetric, that is it does not depend on the variable θ . Let us write the exact solution formulas for the particular case of the Couette flow under consideration:

$$v_{\theta}^*(r^*, z^*) = \frac{(R_3^{*2} - r^{*2})}{r^*(R_3^{*2} - 1)}, \quad v_r^*(r^*, z^*) = 0, \quad v_z^*(r^*, z^*) = 0,$$

$$p^*(r^*, z^*) = p_2^* + (R_3^{*2} - 1)^{-2} \cdot \left[\frac{r^{*2} - 1}{2} - \frac{R_3^{*4}}{2} \left(\frac{1}{r^{*2}} - 1 \right) - 2R_3^{*2} \ln r^* \right], \quad (8)$$

where $p_2^* = p^*(R_2^*, z^*) = p^*(1, z^*)$.

The steady-state solution of the Couette flow problem was obtained by the finite-difference/spectral method described in Section 2 in combination with the steadying method. As a criterion for convergence to the stationary solution, the satisfaction of the following inequality was verified at the end of each time step: $|E^*(t^{n+1}) - E^*(t^n)| < 10^{-4}$, where $E^*(t)$ is the dimensionless kinetic energy of the gas lying in the gap between the cylinders, t^n is the value of time at the n th time level. The exact value E_{th}^* of the kinetic energy was found with *Mathematica* to be

$$E_{th}^* = \frac{1}{2} \int_0^{H^*} \left[\int_1^{R_3^*} \left(\int_0^{2\pi} (v_{\theta}^{*2} + v_r^{*2} + v_z^{*2}) d\theta \right) r dr \right] dz$$

$$= \frac{1}{2} \int_0^{H^*} \left[\int_1^{R_3^*} \left(\int_0^{2\pi} v_{\theta}^{*2} r dr \right) d\theta \right] dz = \frac{4R_3^{*2} - 1 - 3R_3^{*4} + 4R_3^{*4} \ln(R_3^*)}{4(R_3^{*2} - 1)^2} \pi H^*, \quad (9)$$

where H^* is the dimensionless height of the cylinders.

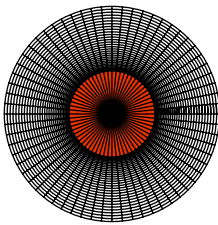


Fig. 3. Computational grid in the (x, y) plane

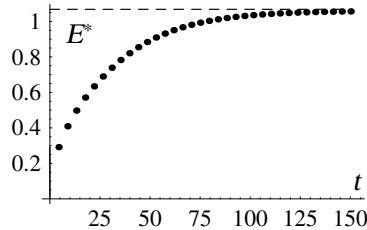


Fig. 4. The dimensionless kinetic energy of gas $E^* = E^*(t^*)$

The computational grid in the interval $[R_2, R_3]$ was nonuniform and was clustered near the walls of cylinders with the use of the function \tanh . The grid was uniform along the θ - and z -axes. Figure 3 shows the grid in the plane of the Cartesian coordinates x, y , where $x = r \cos \theta$, $y = r \sin \theta$. The numerical results presented in Figs. 4 and 5 were obtained on the mesh of 65, 30, and 13 nodes

along the θ -, r -, and z -axes, respectively. The angular velocity of the internal cylinder rotation $\omega = 10$ rpm, the Reynolds number $Re=121.71$, $R_2 = 0.04$ m, $R_3 = 0.10$ m, $H = 0.03$ m.

The dotted line in Fig. 4 shows the graph of the kinetic energy obtained by the finite-difference method described in the foregoing section after the execution of 1700 time steps. The dashed line is the exact stationary value E_{th}^* according to the right-hand side of equality (9). It is seen from Fig. 4 that 1700 time steps proved to be sufficient in the given computational example to reach the steady regime.

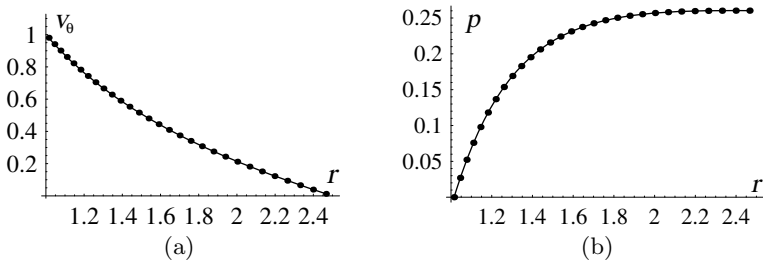


Fig. 5. Profiles of the difference solution (dotted lines) and exact solution (solid lines): (a) the azimuthal velocity v_θ^* ; (b) the pressure p^*

Figure 5 shows the difference solution profiles for v_θ^* and p^* , from which it is seen that the finite-difference method described in Section 2, which was implemented in a FORTRAN code, ensures a high accuracy of the results on a relatively crude grid of $65 \times 30 \times 13$ nodes.

4 Implementation of the Schultz-Grunow Procedure with *Mathematica*

4.1 Boundary Layer of the Casing Base

An approximate analytic solution of the problem of steady flow of the gas or liquid in a closed cylindrical casing was derived in [15]. This flow was induced by

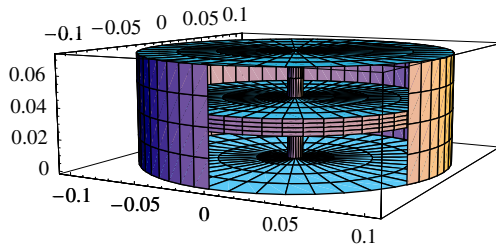


Fig. 6. The closed cylindrical casing with a single internal disc on the shaft

the disc rotation inside the casing. Figure 6 shows the closed cylindrical casing with a disc inside it; a part of the casing front vertical wall has been removed to show the interior.

Following [15] we will assume that the gas flow inside the cylindrical casing is steady and axisymmetric. The latter assumption implies that the derivatives of the solution components with respect to the azimuthal coordinate θ are equal to zero. Therefore, the Navier–Stokes equations (1), (2) take the following form in the case under consideration:

$$u \frac{\partial u}{\partial r} - \frac{v^2}{r} + w \frac{\partial u}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial r} + \nu \left[\frac{\partial^2 u}{\partial r^2} + \frac{\partial}{\partial r} \left(\frac{u}{r} \right) + \frac{\partial^2 u}{\partial z^2} \right], \tag{10}$$

$$u \frac{\partial v}{\partial r} + \frac{uv}{r} + w \frac{\partial v}{\partial z} = \nu \left[\frac{\partial^2 v}{\partial r^2} + \frac{\partial}{\partial r} \left(\frac{v}{r} \right) + \frac{\partial^2 v}{\partial z^2} \right], \tag{11}$$

$$u \frac{\partial w}{\partial r} + w \frac{\partial w}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial z} + \nu \left(\frac{\partial^2 w}{\partial r^2} + \frac{1}{r} \frac{\partial w}{\partial r} + \frac{\partial^2 w}{\partial z^2} \right), \tag{12}$$

$$\frac{\partial u}{\partial r} + \frac{u}{r} + \frac{\partial w}{\partial z} = 0, \tag{13}$$

where $u \equiv v_r$, $v \equiv v_\theta$, $w \equiv v_z$. Following [15,9,10] we now obtain from equations (10)–(13) several useful integral relations. Let us rewrite the left-hand side of equation (10) as

$$u \frac{\partial u}{\partial r} - \frac{v^2}{r} + \frac{\partial(uw)}{\partial z} - u \frac{\partial w}{\partial z}. \tag{14}$$

Let us multiply the both sides of the continuity equation (13) by u : $u \frac{\partial u}{\partial r} + \frac{u^2}{r} + u \frac{\partial w}{\partial z} = 0$, from where we have

$$u \frac{\partial w}{\partial z} = -u \frac{\partial u}{\partial r} - \frac{u^2}{r}. \tag{15}$$

Replacing the term $u \partial w / \partial z$ in (14) in accordance with formula (15) we obtain the equality

$$u \frac{\partial u}{\partial r} - \frac{v^2}{r} + \frac{\partial(uw)}{\partial z} - u \frac{\partial w}{\partial z} = \frac{\partial(u^2)}{\partial r} + \frac{u^2}{r} - \frac{v^2}{r} + \frac{\partial(uw)}{\partial z}. \tag{16}$$

Let us denote by $\delta_0(r)$ the thickness of the boundary layer on the motionless base of the casing. Let us replace the left-hand side of equation (10) with the right-hand side of equation (16), then multiply the both sides of the obtained equation by $r dz$ and integrate over z from zero to $\delta_0(r)$. We take into account the fact that

$$\int_0^{\delta_0(r)} r \frac{\partial}{\partial r} (u^2) dz = \int_0^{\delta_0} \frac{\partial}{\partial r} (ru^2) dz - \int_0^{\delta_0} u^2 dz. \tag{17}$$

Following [15] we further assume that there is no radial motion of gas at the external edge of the boundary layer $z = \delta_0(r)$, that is $u = 0$ at $z = \delta_0$. Besides, due to the no-slip condition on the casing motionless base, we have that $u = 0$, $w = 0$ at $z = 0$. With regard for these conditions and equation (17) we obtain from (10) the first integral relation:

$$d/dr \left(r \int_0^{\delta_0} u^2 dz \right) - \int_0^{\delta_0} v^2 dz = -\nu r (\partial u / \partial z)_{z=0} - (r/\rho) \int_0^{\delta_0} (\partial p / \partial r) dz. \tag{18}$$

We now rewrite the left-hand side of equation (11) with regard for (13) in the form $\frac{\partial}{\partial r}(uv) + \frac{\partial}{\partial z}(vw) + 2\frac{uv}{r}$ and, multiplying the both sides of (11) by $r^2 dz$, integrate over z from zero to $\delta_0(r)$. Using (13) and integrating by parts we obtain the second integral relation

$$d/dr \left(r^2 \int_0^{\delta_0(r)} uv dz \right) - \tilde{\omega} r^2 (d/dr) \left(r \int_0^{\delta_0} u dz \right) = -\nu r^2 (\partial v / \partial z)_{z=0}. \tag{19}$$

We have used here the boundary condition at the boundary layer edge $z = \delta_0(r)$: $v(r, z) = \tilde{\omega} r$, where $\tilde{\omega}$ is a constant angular velocity of the rotation of the external flow in the gap between the boundary layers as a quasi-solid body [15,9,10], this quantity will be determined below from the condition of the equality of the moment of friction forces in the liquid on the disc surface to the viscous friction force moments on the casing butt ends and on the casing lateral wall.

The following velocity profiles were taken in [15] as the approximate ones:

$$u = -u_0(r) \left[1 - \left(2\frac{z}{\delta_0(r)} - 1 \right)^2 \right], \quad v = r\tilde{\omega} \left[1 - \left(1 - \frac{z}{\delta_0(r)} \right)^2 \right], \tag{20}$$

where $u_0(r)$ is the maximum radial velocity inside the boundary layer, which is unknown for a while. and the minus sign affecting it is taken because the radial velocity component on the casing base is directed from the periphery to the axis.

Let us substitute the representations (20) for u and v into the integrals entering the left-hand sides of equations (18) and (19), and let us calculate these integrals with the aid of CAS *Mathematica*:

```
u = -u0*(1 - (2z/del0 - 1)^2); v = r*t*(1 - z/del0)^2;
itgr1 = Integrate[u^2,{z,0,del0}]; itgr2 = Integrate[v^2,{z,0,del0}];
itgr3 = Integrate[u*v,{z,0,del0}]; itgr4 = Integrate[u,{z,0,del0}]
```

Here and in the following, $\tau = \tilde{\omega}$. The CAS *Mathematica* has produced the following results for the integrals:

$$\int_0^{\delta_0} u^2(r, z) dz = (8/15)\delta_0 u_0^2; \quad \int_0^{\delta_0} v^2(r, z) dz = (8/15)\delta_0 r^2 \tilde{\omega}^2; \\ \int_0^{\delta_0} u(r, z)v(r, z) dz = -(7/15)\delta_0 r \tilde{\omega} u_0; \quad \int_0^{\delta_0} u(r, z) dz = -(2/3)\delta_0 u_0. \tag{21}$$

According to (18) and (19), we will also need the expressions for the first derivatives of functions (20) with respect to z at $z = 0$. Let us calculate them with the aid of the CAS *Mathematica*:

```
duz0 = D[u,z]/.z -> 0; dvz0 = D[v,z]/.z-> 0
```

$$(\partial u / \partial z)_{z=0} = -4u_0 / \delta_0; \quad (\partial v / \partial z)_{z=0} = 2r\tilde{\omega} / \delta_0. \tag{22}$$

We obtain for the derivative $\partial p/\partial r$ in the external flow the following equality from equation (10):

$$\partial p/\partial r = \rho v^2/r. \tag{23}$$

On the other hand, relation $v(r, z) = \tilde{\omega}r$ holds in the external flow, therefore, we obtain from (23): $\partial p/\partial r = \rho\tilde{\omega}^2r$. This derivative and the integral of it enter equation (18), therefore, it is also necessary to insert in the *Mathematica* program the corresponding symbolic computations:

```
dpdr = rho*t^2*r; itgrp = Integrate[dpdr, {z, 0, de10}];
```

$$\int_0^{\delta_0} (\partial p/\partial r) dz = \delta_0 r \tilde{\omega}^2 \rho.$$

Using the integrals calculated above we are now ready to obtain from the first integral relation (18) the first ordinary differential equation. Following [15,9,10] we introduce the function $\Phi(r)$: $\Phi(r) = u_0(r)\delta_0(r)$. It is convenient to account for the variability of quantity δ_0 with the aid of the following transformation rules for integrals (21):

```
ruledu0 = {de10 -> de11[r], u0 -> Phi[r]/de11[r]};
itgr1v = itgr1/.ruledu0; itgr2v = itgr2/.ruledu0
itgr3v = itgr3/.ruledu0; itgr4v = itgr4/.ruledu0
```

It is now easy to compute the left-hand side of equation (18):

```
eq1 = D[r*itgr1v, r] - itgr2v + nu*r*dudz0 + r/rho*itgrp
```

As a result we obtain the left-hand side of the first ordinary differential equation:

$$\frac{7}{15}\delta_0 r^2 \tilde{\omega}^2 - \frac{4r u_0 \nu}{\delta_0^2} + \frac{8\Phi^2(r)}{15\delta_0(r)} - \frac{8r\Phi^2(r)\delta_0'(r)}{15\delta_0^2(r)} + \frac{16r\Phi(r)\Phi'(r)}{15\delta_0(r)} \tag{24}$$

If we multiply this expression by 15/8 then we can easily see that the expression will be obtained, which coincides with the left-hand side of the first equation of system (11) from [15]. The prime by $\delta_0(r)$ and $\Phi(r)$ in (24) means the differentiation with respect to r . To reduce the amount of further symbolic computations it is desirable to get rid of rational coefficients in (24). This was done with the aid of the following *Mathematica*-commands:

```
eq1 = Expand[Simplify[15*de11[r]^2*eq1]]; Print["eq1 = ", eq1]
eq1 = 7r^2*omega^2*delta_0^3 - 60r*nu*Phi(r) + 8*delta_0(r)*Phi^2(r) - 8r*Phi^2(r)*delta_0'(r) + 16r*delta_0(r)*Phi(r)*Phi'(r). \tag{25}
```

The computation of the left-hand side of the second ordinary differential equation by the substitution of formulas (21) and (22) into (19) is carried out similarly with *Mathematica*:

```
eq2 = Simplify[D[r^2*itgr3v, r] - t*r^2*D[r*itgr4v, r] + nu*r^2*dvz0];
eq2 = Simplify[eq2]; eq2 = Expand[15de11[r]*eq2/(t*r^2)];
Print["eq2 = ", eq2];
eq2 = 30r*nu - 11*delta_0(r)*Phi(r) + 3r*delta_0(r)*Phi'(r). \tag{26}
```

We further tried to apply the *Mathematica*-function `DSolve[...]` for finding the solution of system $eq1 = 0, eq2 = 0$. This function was unfortunately unable to find the exact general solution of this nonlinear system. In this connection, we apply in the following an approximate procedure described in [15], that is we introduce the new independent variable $x = 1 - (r/b)$, where b is the casing radius. Let us introduce the notations: $\bar{\Phi}(x) = \Phi(b - bx)$, $\bar{\delta}_0(x) = \delta_0(b - bx)$. We now derive the left-hand sides of the both ordinary differential equations for $\bar{\Phi}(x)$ and $\bar{\delta}_0(x)$ by applying the relations $\frac{d\Phi(r)}{dr} = \frac{d\bar{\Phi}(x)}{dx} \cdot \frac{dx}{dr} = -\frac{1}{b} \frac{d\bar{\Phi}(x)}{dx}$, $\frac{d\delta_0(r)}{dr} = -\frac{1}{b} \frac{d\bar{\delta}_0(x)}{dx}$. This was realized in the *Mathematica* program as follows (we present the program fragment only for the transformation of the first equation, the transformation of the second equation is performed in the same way):

```

eq1 = eq1/.r -> b*(1 - x)
eq1 = eq1/. {del1[b (1 - x)] -> del1[x], del1'[b(1 - x)] -> -del1'[x]/b};
eq1 = eq1/. {Phi[b*(1 - x)] -> Phi[x], Phi'[b*(1 - x)] -> -Phi'[x]/b}
eq1t = eq1/. {del1[x] -> delta_0[x], del1'[x] -> delta_0'[x], t -> w};
Print["eq1 = ", TraditionalForm[eq1t]];

eq1 = 7b^2*w^2(1 - x)^2*delta_0^3(x) + 8Phi^2(x)*delta_0(x) - 16(1 - x)*Phi(x)*Phi'(x)*delta_0(x)
      - 60b(1 - x)*nu*Phi(x) + 8(1 - x)*Phi^2(x)*delta_0'(x);
eq2 = 30b(1 - x)*nu - 11Phi(x)*delta_0(x) - 3(1 - x)*delta_0(x)*Phi'(x).

```

Here and in the following, the bars over $\bar{\Phi}(x)$ and $\bar{\delta}_0(x)$ are omitted for the sake of notation brevity.

Following [15], we now search for the solution of system $eq1 = 0, eq2 = 0$ in the form of the series

$$\Phi = x^n(c_0 + c_1x + c_2x^2 + \dots), \quad \delta_0 = x^p(d_0 + d_1x + d_2x^2 + \dots). \tag{28}$$

The exponents n and p in (28) were then chosen in [15] from the requirement that the exponents in the system $eq1 = 0, eq2 = 0$ differed from one another by unity; this gives $n = 3/4, p = 1/4$. It was observed in [15] that if one specifies expressions (28) in the form

$$\Phi = x^{\frac{3}{4}}\sqrt{\nu\tilde{\omega}}(c_0 + c_1x + c_2x^2 + \dots), \quad \delta_0 = x^{\frac{1}{4}}\sqrt{\frac{\nu}{\tilde{\omega}}}(d_0 + d_1x + d_2x^2 + \dots), \tag{29}$$

then one obtains for determining the coefficients c_j and d_j the polynomial equations whose coefficients depend only on c_j and d_j . The case was considered in [15] when Φ and δ_0 involve the fourth-degree polynomials in x :

$$\Phi(x) = x^{3/4}b\sqrt{\nu\tilde{\omega}}P_1(x), \quad \delta_0(x) = x^{1/4}\sqrt{\nu/\tilde{\omega}}P_2(x), \tag{30}$$

where

$$P_1(x) = \sum_{j=0}^{N_1} c_j x^j, \quad P_2(x) = \sum_{j=0}^{N_2} d_j x^j, \tag{31}$$

$N_1 = N_2 = 4$. The substitution of (30) and (31) in the left-hand sides (27) was implemented by us with *Mathematica* as follows:

```

P1 = c0 + c1*x + c2*x^2 + c3*x^3 + c4*x^4;
P2 = d0 + d1*x + d2*x^2 + d3*x^3 + d4*x^4;
Fi = b*Sqrt[t*nu]*x^(3/4)*P1;
del1 = Sqrt[nu/t]*x^(1/4)*P2;
dfix= D[Fi,x]; eq2r = eq2/. {Phi[x] -> Fi, Phi'[x] -> dfix, del1[x] -> del2}
eq2r = Expand[4eq2r/b]; eq2r = eq2r/. Sqrt[x/t] Sqrt[t*nu] -> nu;
eq2r = Expand[Simplify[eq2r/nu]]; Print["eq2r = ",eq2r];
d-del12x = D[del2, x];
eq1n=eq1/.{del1[x]-> del2, Phi[x] -> Fi, Phi'[x] -> dfix, del1'[x] -> d-del12x}
eq1r = Expand[x^(-3/4)*eq1n/b^2];
eq1r = Expand[PowerExpand[Simplify[eq1r*Sqrt[t/nu]]]];
eq1r = Expand[Simplify[eq1r/nu/t]]; Print["eq1r = ",eq1r];

```

As a result of these symbolic computations, equations $eq1 = 0$, $eq2 = 0$ have reverted into the following polynomial equations:

$$\sum_{k=0}^{M_1} A_k(\mathbf{X})x^k = 0, \quad \sum_{k=0}^{M_2} B_k(\mathbf{X})x^k = 0, \tag{32}$$

where $\mathbf{X} = (c_0, c_1, c_2, c_3, c_4, d_0, d_1, d_2, d_3, d_4)$, $M_1 = 14$, $M_2 = 9$. It is easy to obtain the expressions for the coefficients $A_k(\mathbf{X})$ and $B_k(\mathbf{X})$ with the aid of the following *Mathematica* commands:

```

pol = eq1r; np1 = Exponent[eq1r, x]; np2 = Exponent[eq2r, x];
Print["np1 = ", np1, "; np2 = ", np2];
A = {eq1r/.x -> 0}; B = {eq2r/. x -> 0}; Print["A(0) = ",A[[1]]];
Do[aj = Coefficient[pol,x^j]; AppendTo[A,aj];
Print["A(",j,") = ",aj], {j, np1}];
pol = eq2r; Print["B(0) = ",B[[1]]];
Do[aj = Coefficient[pol,x^j]; AppendTo[B,aj];
Print["B(",j,") = ",aj], {j, np2}];

```

As a result we obtain the expressions for A_k and B_k in (32). We present below the formulas for A_k and B_k only for $k = 0, 1$.

$$\begin{aligned}
A_0 &= -60c_0 - 10c_0^2d_0 + 7d_0^3; \\
A_1 &= 60c_0 - 60c_1 + 18c_0^2d_0 - 36c_0c_1d_0 - 14d_0^3 - 2c_0^2d_1 + 21d_0^2d_1; \\
B_0 &= 120 - 9c_0d_0; \quad B_1 = -120 - 35c_0d_0 - 21c_1d_0 - 9c_0d_1.
\end{aligned}$$

The coefficients c_j, d_j in (31) were computed in [15] from the requirement that the both equations (32) are identically equal to zero. This means that the following systems of polynomial equations were solved:

$$A_k(\mathbf{X}) = 0, \quad k = 0, \dots, M_1; \quad B_k(\mathbf{X}) = 0, \quad k = 0, \dots, M_2. \tag{33}$$

As a result, the numerical values of the coefficients c_j, d_j in (29) were found in [15] so that

$$\Phi = x^{3/4}b\sqrt{\nu\tilde{\omega}}(3.04 - 4.64x + 2.839x^2 - 2.855x^3 - 1.814x^4); \tag{34}$$

$$\delta_0 = x^{1/4}\sqrt{\nu/\tilde{\omega}}(4.385 - 5.845x + 4.015x^2 - 4.46x^3 - 1.29x^4). \tag{35}$$

However, the computation of the numerical values of the coefficients A_k and B_k with the numerical values of c_j and d_j found in [15] shows that they are different from zero:

```
ruleSG = {c0-> 3.04, c1-> -4.64, c2-> 2.839, c3 -> -2.855, c4 -> -1.814,
d0 -> 4.385, d1 -> -5.845, d2 -> 4.015, d3 -> -4.46, d4 -> -1.29};
Do[aj = A[[j]]/.ruleSG; j0= j- 1; Print["ASG(",j0,") = ",aj],{j,np1+1}];
Do[aj = B[[j]]/.ruleSG; j0= j- 1; Print["BSG(",j0,") = ",aj],{j,np2+1}];

A0 = 2.5662, A1 = -15.6054, A2 = 23.6544, ..., A7 = -14429.8, ..., A14 = -15.0268;
B0 = 0.0264, B1 = 0.6296, ..., B7 = -668.551, B8 = -24.5247, B9 = 30.4208.
```

We will need in the following a formula for the moment M_1 of friction forces of the fluid on two motionless butt-end casing walls each of which is a disc of radius $b = R_3$ in order to determine the angular velocity $\tilde{\omega}$. It is assumed here and in the following that the disk lies in the casing middle. According to [15] and with regard for (22) and (30), the above total moment for the both butt ends is expressed by the formula

$$\begin{aligned}
 M_1 &= -2 \int_0^b 2\pi r^2 \mu \left(\frac{\partial v}{\partial z} \right)_{z=0} \cdot dr = -8\pi\mu\tilde{\omega}b^4 \int_0^1 \frac{(1-x)^3}{\delta_0(x)} dx \\
 &= -8\pi\mu\tilde{\omega}b^4 \sqrt{\frac{\tilde{\omega}}{\nu}} \int_0^1 \frac{(1-x)^3 x^{-1/4}}{P_2(x)} dx.
 \end{aligned}
 \tag{36}$$

At the numerical computation of the integral on the right-hand side of (36) it was accounted for the fact that $P_2(x)$ has the root $x^* = 0.746836$. As a result, the numerical value of the integral $8\pi \int_0^1 (1-x)^3 x^{-1/4} / P_2(x) dx$ was found with the aid of the *Mathematica* function `NIntegrate[...]` to be equal to 3.4048249. On the other hand, the value of the integral under consideration was found by a graphical technique in [15] and amounted to 3.387.

4.2 The Boundary Layer of the Disc

Let us now proceed to the consideration of the boundary layer of the disc rotating with angular velocity ω . Following [15] we assume the presence of the constant angular velocity $\tilde{\omega} < \omega$ in the co-flow of the external liquid, which rotates as a quasi-solid body. Denote as in [15] by s the distance between the motionless casing base and the lower horizontal disc surface. Following [15] let us introduce the coordinate $\bar{z} = s - z$, so that $\bar{z} = 0$ on the lower horizontal surface of the disc. We now specify as in [15] the following polynomial representations for u and v near the disc surface:

$$u = u_0^* \left[1 - \left(2\frac{\bar{z}}{\delta} - 1 \right)^2 \right], \quad v = \omega r - (\omega - \tilde{\omega})r \left[1 - \left(1 - \frac{\bar{z}}{\delta} \right)^2 \right],
 \tag{37}$$

where u_0^* is a new unknown maximum radial velocity, $\delta(r)$ is the thickness of the boundary layer of the disc. We omit in the following the bar over z for the sake of brevity. The integral relations (18) and (19) retain the same form, one must only

replace δ_0 with δ . Similarly to the foregoing we at first calculate in symbolic form the integrals and the derivatives entering (18) and (19). Substituting the found expressions into (18) and (19) we obtain the system of two ordinary differential equations for $\delta(r)$ and $u_0^*(r)$:

$$-\frac{1}{5}\omega^2 r^2 \delta - \frac{4}{15}\omega r^2 \tilde{\omega} \delta + \frac{7}{15}r^2 \tilde{\omega}^2 \delta + \frac{4r\nu u_0^*}{\delta} + \frac{8}{15}\delta u_0^{*2} + \frac{8}{15}r u_0^{*2} \delta' + \frac{16}{15}r \delta u_0^* u_0^{*'} = 0;$$

$$-\frac{2\omega r^3 \nu}{\delta} + \frac{2r^3 \tilde{\omega} \nu}{\delta} + \frac{3}{5}\omega r^2 \delta u_0^* + \frac{11}{15}r^2 \tilde{\omega} \delta u_0^* + \frac{1}{5}\omega r^3 u_0^{*'} \delta' - \frac{1}{5}r^3 \tilde{\omega} u_0^{*'} \delta'$$
(38)

$$+\frac{1}{5}\omega r^3 \delta u_0^{*'} - \frac{1}{5}r^3 \tilde{\omega} \delta u_0^{*'} = 0.$$
(39)

Unlike the case of the boundary layer of the motionless casing base, the system (38), (39) has a particular solution of the form $\delta(r) = c_1, u_0^* = c_2 r$, where c_1 and c_2 are constants. For the sake of brevity, we omit the corresponding fragment of our *Mathematica* program and present the obtained nonlinear algebraic equations for the determination of c_1, c_2 :

$$6c_1^2 c_2^2 - (3/4)c_1^2 \omega^2 - c_1^2 \omega \tilde{\omega} + (7/4)c_1^2 \tilde{\omega}^2 + 15c_2 \nu = 0;$$
(40)

$$(4/5)c_1^2 c_2 \omega + (8/15)c_1^2 c_2 \tilde{\omega} - 2\nu(\omega - \tilde{\omega}) = 0.$$
(41)

Equation (41) is linear in c_2 , therefore, finding c_2 from (41), we substitute the found expression in (40) and find c_1 :

$$c_2 = \frac{15\nu(\omega - \tilde{\omega})}{2c_1^2(3\omega + 2\tilde{\omega})}; \quad c_1 = \sqrt{\frac{\nu}{\tilde{\omega}}} \cdot \frac{\sqrt[4]{150}}{\sqrt{2 + 3\frac{\omega}{\tilde{\omega}}}} \sqrt[4]{\frac{6\frac{\omega}{\tilde{\omega}} - 1}{\frac{\omega}{\tilde{\omega}} + \frac{7}{3}}}.$$
(42)

This result coincides with the one obtained in [15]. Let a be the disc radius. The friction moment on the rotating disc is [15,9,10]

$$M_2 = -\int_0^a 4\pi\mu(\omega - \tilde{\omega})2\frac{r^3}{\delta} dr = -\frac{2\pi}{\sqrt[4]{150}}\mu a^4(\omega - \tilde{\omega})\sqrt{\frac{\tilde{\omega}}{\nu}} \cdot \sqrt{2 + 3\frac{\omega}{\tilde{\omega}}} \cdot \sqrt[4]{\frac{\omega}{6\frac{\omega}{\tilde{\omega}} - 1} + \frac{7}{3}}.$$
(43)

It remains to determine the unknown angular velocity $\tilde{\omega}$ of the fluid rotation in the casing far from solid walls. To this end, the requirement of the equality of two moments was used in [15,9,10]: the moment M_1 decelerating the fluid on the butt-end walls, which is determined by formula (36), and the moment M_2 (43), which decelerates the disc. But the radius of the cylindric base b enters the expression for M_1 . It was assumed in [15] that the gap between the disc and the casing wall is small. Then $b \approx a$, and one can resolve equation $M_1 = M_2$ with respect to $\tilde{\omega}$:

```
eqM = 3.404825 - 2Pi / (150.^(1/4)) * (zet - 1) * Sqrt[2 + 3zet] * ((zet + 7/3) / (6zet - 1))^(1/4)
sol = NSolve[eqM == 0, zet]; z0 = zet /. sol[[1]]; z2 = 1/z0
```

We have introduced here the notation $\mathbf{zet} = \omega/\tilde{\omega}$. As a result, it turned out that $\omega/\tilde{\omega} = 1.89321$ so that $\tilde{\omega}/\omega = 0.528202$. According to [15,9,10], $\tilde{\omega}/\omega = 0.54$, so

that the quasi-solid kernel between the boundary layers rotates according to [15,9,10] at about halved angular velocity in comparison with the disc.

It is convenient for the following to introduce two dimensionless parameters: the Reynolds number $Re = R_2^2 \omega / \nu$ and the number $\beta = \tilde{\omega} / \omega$; $\beta = 0.54$ in the case of the solution (34), (35) from [15]. The formula for the thickness $\delta(r)$ of the boundary layer of the horizontal surface of the rotating disc may then be written as

$$\delta = \frac{R_2}{\sqrt{Re}} \cdot \frac{\sqrt[4]{150}}{\sqrt{2\beta + 3}} \cdot \sqrt[4]{\frac{6 - \beta}{1 + \frac{7}{3}\beta}}. \tag{44}$$

We show in Fig. 8 the graphs of the functions $z = \delta_0(r)$ and $z = \delta(r)$ for the case when $Re = 1288$, $R_2 = 0.092$ m, $s = 0.031$ m, $\omega = 20$ rpm in (35) and (44). As can be seen in Fig. 8, the lower and upper boundary layers do not intersect. The thickness $\delta_0(r)$ described by formula (35) is negative in the shaft neighborhood. This contradicts the physical meaning of the boundary-layer thickness: it should always be non-negative.

5 A New Solution for $N_1 = N_2 = 4$

Let us identify several shortcomings of the Schultz-Grunow’s solution (see formulas (34) and (35)).

(i) Near the shaft, the thickness $\delta_0(r)$ of the boundary layer of the casing base is negative according to (35). This is seen in Fig. 8 and can also be easily proved mathematically as follows. Let us denote by R_1, R_2 , and R_3 the radii of the shaft, disc, and the cylindrical lateral casing wall. For real disc machines, the inequality $R_1 \ll R_3$ is satisfied. Then the corresponding value $x = 1 - (r/R_3) \approx 1$ on the shaft surface. Substituting the value $x = 1$ into (35) it is easy to see that $\delta_0(1) < 0$.

(ii) Formulas (34) and (35) do not ensure the satisfaction of the no-slip condition $u(R_1, z) = 0$ on the shaft surface. Let us indeed consider the formula $u_0(r) = \Phi(r)/\delta_0(r)$. The quantity $u_0(r)$ can vanish at $r = R_1$ if and only if $\Phi(1 - R_1/R_3) = 0$. Since $x \approx 1$ at $r = R_1$, we obtain from (34) that in order to ensure the equality $\Phi = 0$ at $r = R_1$ it is necessary that $P_1(1) = 0$. But $P_1(1) = -3.43 \neq 0$ according to (34).

(iii) Formulas (34) and (35) do not ensure the satisfaction of the no-slip condition $u = v = 0$ on the motionless vertical circular wall of the casing ($r = R_3$), see also Fig. 7.

(iv) The procedure for determining the angular velocity $\tilde{\omega}$ of a quasi-solid fluid rotation between the lower and upper boundary layers described in [15,9,10] does not account for the viscous friction force moment on a circular lateral casing wall.

Below in the present section, we will construct such a solution of the problem under consideration, in which the above shortcomings are eliminated. We

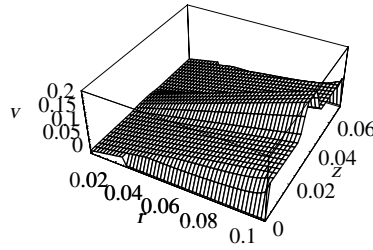


Fig. 7. Surface $v = v(r, z)$ in accordance with the Schultz–Grunow’s solution [15]

begin with eliminating the shortcoming (ii). It is easy to account for the no-slip condition $\Phi(1 - R_1R_3^{-1}) = 0$ by specifying Φ in the form

$$\Phi(x) = x^{3/4}b\sqrt{\nu\bar{\omega}}(1 - R_1/R_3 - x)\bar{P}_1(x), \tag{45}$$

where

$$\bar{P}_1(x) = \sum_{j=0}^{N_1-1} c_j x^j. \tag{46}$$

Representation (45) has a shortcoming at its practical use: the coefficients c_j and d_j will now depend on a specific value of the ratio $\varepsilon = R_1/R_3$. We will, therefore, proceed as follows: let us assume that the ratio R_1/R_3 is small, and we neglect it by specifying $\Phi(x)$ in the form

$$\Phi(x) = x^{3/4}b\sqrt{\nu\bar{\omega}}(1 - x)\bar{P}_1(x). \tag{47}$$

Substituting expression (47) and the expression for δ_0 from (30) into the left-hand sides of equations (27) we obtain a system of 25 polynomial equations (33) for nine unknowns $c_0, c_1, c_2, c_3, d_0, d_1, d_2, d_3, d_4$. Thus, (33) is the overdetermined polynomial system. A widely accepted method for solving such systems is based on the method of least squares [2]. Let us introduce the objective function $F(\mathbf{X})$ by the formula

$$F(\mathbf{X}) = \sum_{k=0}^{M_1} [A_k(\mathbf{X})]^2 + \sum_{k=0}^{M_2} [B_k(\mathbf{X})]^2. \tag{48}$$

The solution \mathbf{X}^* of system (33) is then sought for from the requirement of the minimization of function (48): $F(\mathbf{X}) \rightarrow \min$. Note that on the numerical values c_j, d_j entering (34) and (35), the function $F(\mathbf{X})$ takes the value $F = 6.0625 \cdot 10^8$.

To find the minimum of function (48) we have applied the *Mathematica* function `FindMinimum[...]`:

```
FindMinimum[Funp, {c0, 3.04}, {c1, -4.64}, {c3, -2.855}, {c4, -1.814},
{d0, 4.385}, {d1, -5.845}, {d2, 4.015}, {d3, -4.46}, {d4, -1, 29},
MaxIterations->175, WorkingPrecision -> 20]
```

As the initial guess for the point of the minimum of $F(\mathbf{X})$ we have taken the values of c_j, d_j from (34) and (35). It turned out that 30 iterations used by

default in the method of numerical minimization implemented in the function `FindMinimum[...]` are not enough to determine the point of the minimum \mathbf{X}^* with the required accuracy. Besides, we had to increase the working precision of computations from the default precision 16 to precision 20.

Table 1. Convergence dynamics of the function (48)

N_{it}	30	100	150	175
F^*	$1.405 \cdot 10^5$	$1.722 \cdot 10^4$	$4.513 \cdot 10^3$	$4.513 \cdot 10^3$

In Table 1, N_{it} denotes the user-specified number of iterations, $F^* = F(\mathbf{X}^*)$. It can be seen that the process of the numerical minimization has converged already by 150 iterations. Further iterations have not led to the alteration of F^* . It is also seen in Table 1 that the final value of F^* obtained after 150 iterations is by about five decimal orders lower than the value of F obtained on those values of c_j and d_j , which were computed in [15]. As a result of executed computations, the numerical values of c_j, d_j were found, which provide the minimum of function (48) so that the corresponding approximate analytic solution for $\Phi(x)$ and $\delta_0(x)$ now has the form

$$\Phi_{4,4} = x^{3/4} b \sqrt{\nu \tilde{\omega}} (1 - x) (2.703488 - 1.644563x + 0.909018x^2 + 0.021419x^3); \tag{49}$$

$$\delta_{0,4,4} = x^{1/4} \sqrt{\nu / \tilde{\omega}} (4.064651 - 5.198975x + 4.005359x^2 - 0.563374x^3 - 0.047782x^4). \tag{50}$$

The subscripts 4,4 by Φ and δ_0 in (49) and (50) point to the fact that the degrees N_1 and N_2 of the employed approximating polynomials $P_1(x)$ and $P_2(x)$ in (31) are $N_1 = 4$ and $N_2 = 4$.

Although the ratio R_1/R_3 is assumed small, nevertheless at $r = R_1$ the factor $1 - x = R_1/R_3 \neq 0$. Assume that the coefficients c_j depend weakly on $\varepsilon = R_1/R_3$. We can then retain in the formula for $\Phi_{4,4}$ those numerical values of coefficients c_j , which were found in the limit as $\varepsilon \rightarrow 0$, and replace the factor $1 - x$ with the factor $1 - \frac{R_1}{R_3} - x$ for the purpose of the exact satisfaction of the no-slip condition $u(R_1, z) = 0$:

$$\Phi_{4,4} = x^{3/4} b \sqrt{\nu \tilde{\omega}} \left(1 - \frac{R_1}{R_3} - x\right) (2.703488 - 1.644563x + 0.909018x^2 + 0.021419x^3). \tag{51}$$

It is easy to see from (50) that $\delta_{0,4,4}(1) > 0$, that is we have eliminated also the shortcoming (i). We believe that this shortcoming has been eliminated owing to the solution of the problem of minimizing function (48) with machine accuracy.

As can be seen in Fig. 8, the boundary layer thickness $\delta_{0,4,4}(r)$ (see formula (50)) on the motionless casing base slightly reduces with the decreasing distance r from the shaft surface.

Let us proceed to the elimination of shortcoming (iii). As can be seen from the approximate analytic representation (37) of the solution for the boundary layer

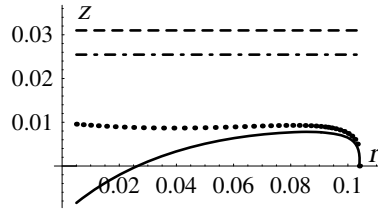


Fig. 8. The curves $z = \delta(r)$ by different formulas: solid line is $z = \delta_0(r)$ according to [15], the dotted line is $z = \delta_{0,4,4}(r)$, the dashed line is $z = s$, the dash-dot line is $z = s - \delta(r)$

of the disc, it uses the boundary condition $v = \omega r$ on the disc surface. But this condition is applicable only in the interval $[R_1, R_2]$. Let $\mathbf{U}_d = (u(r, z), v(r, z))$ be the vector of the approximate analytic solution in the interval $[R_1, R_2]$, that is we restrict the domain of the applicability of this solution by the given interval. Let us expand the solution $\mathbf{U} = (u, v)$ into the Taylor series in the interval $[R_2, R_3]$ with respect to point $r = R_2$:

$$\mathbf{U} = \mathbf{U}(R_2, z) + \frac{\partial \mathbf{U}(R_2, z)}{\partial r}(r - R_2) + O[(r - R_2)^2], \quad r \in [R_2, R_3]. \quad (52)$$

It follows from (52) that at small values of the gap between the disc and the casing (which is typical of disc fans) the linear approximation

$$\mathbf{U}(r, z) = \mathbf{A}r + \mathbf{B} \quad (53)$$

may be an approximation, which is satisfactory in terms of accuracy. It is easy to find the expressions for the constant vectors \mathbf{A} and \mathbf{B} in (53) with regard for conditions $\mathbf{U}(R_3, z) = (0, 0)$, so that we obtain the following formula for the approximate analytic solution $\mathbf{U}(r, z)$ in the interval $[R_1, R_3]$:

$$\mathbf{U}(r, z) = \begin{cases} \mathbf{U}_d(r, z), & R_1 \leq r \leq R_2; \\ \mathbf{U}_d(R_2, z) \cdot (r - R_3)/(R_2 - R_3), & R_2 \leq r \leq R_3. \end{cases} \quad (54)$$

It is easy to see that the solution \mathbf{U} determined by formula (54) satisfies the no-slip condition $\mathbf{U}(R_3, z) = (0, 0)$.

Let us now find the expression for the viscous friction force moment on the circular lateral wall of the casing for the purpose of eliminating the shortcoming (iv) of the solution from [15]. Let us take a strip of height dz on the cylinder surface. The viscous friction force moment equals $2\pi b dz \cdot b\mu(\partial v/\partial r)|_{r=b}$ on this elementary strip. The integral moment M_3 of the viscous friction force on the surface of the cylinder of height H is $M_3 = \int_0^H \varphi(r, z) dz$, where $\varphi(r, z) = 2\pi b^2 \mu \cdot (\partial v/\partial r)$. To account for the presence of the boundary layers and the disc in the interval $0 \leq z \leq H$ let us partition this interval into several subintervals:

$$M_3 = \int_0^{s-\delta} \varphi dz + \int_{s-\delta}^s \varphi dz + \int_s^{s+d} \varphi dz + \int_{s+d}^{s+d+\delta} \varphi dz + \int_{s+d+\delta}^H \varphi dz,$$

where d is the disc thickness. In the interval $(0, s - \delta)$ we have according to (54) the expression $v(r, z) = \tilde{\omega}R_2(r - R_3)/(R_2 - R_3)$, $R_2 \leq r \leq R_3$, therefore, $\partial v/\partial r = \tilde{\omega}R_2/(R_2 - R_3)$,

$$\int_0^{s-\delta} \varphi dz + \int_{s+d+\delta}^H \varphi dz = -4\pi b^2 \mu \tilde{\omega} (s - \delta) R_2 / (R_3 - R_2). \tag{55}$$

We further obtain with regard for formulas (37) and (54) that

$$\int_{s-\delta}^s \varphi dz + \int_{s+d}^{s+d+\delta} \varphi dz = -(4/3)\pi b^2 \mu \delta (\omega + 2\tilde{\omega}) R_2 / (R_3 - R_2). \tag{56}$$

And, finally, at the calculation of the remaining integral in the interval $[s, s + d]$ we assume that in the gap $[R_2, R_3]$ in front of the disc the gas flow differs little from the Couette flow. Replacing R_1 with R_2 in formula (8) and calculating $\partial v/\partial r$ from (8) we obtain:

$$\int_s^{s+d} \varphi dz = -2\pi b^2 \mu \omega R_2 \cdot 2R_3 d / (R_3^2 - R_2^2). \tag{57}$$

The quantity $\tilde{\omega}$ was found in [15] by solving the equation $M_2 = M_1$, where the expressions for M_1 and M_2 are given by formulas (36) and (43). We now take into account also the moment M_3 and will determine $\tilde{\omega}$ from the equation

$$M_2 = M_1 + M_3. \tag{58}$$

Substituting into (58) the expressions (36), (43), (55)–(57) and multiplying the both sides of the obtained equality by $\lambda_1(\zeta/\text{Re})^{1/2}/(\mu b^3 \tilde{\omega})$, where $\lambda_1 = R_2/b$, we arrive at the equation

$$\begin{aligned} & \frac{2\pi\lambda_1^4}{\sqrt[4]{150}}(\zeta - 1)\sqrt{2 + 3\zeta} \sqrt[4]{\frac{\zeta + \frac{7}{3}}{6\zeta - 1}} = I_1 + 4\pi\sqrt{\frac{\zeta}{\text{Re}}}\lambda_1[\lambda_2 - \lambda_1\bar{\delta} \\ & + \frac{1}{3}\lambda_1\bar{\delta}(\zeta + 2)]\frac{\lambda_1}{1-\lambda_1} + 4\pi\lambda_1^2\lambda_3\zeta^{3/2}/[(1 - \lambda_1^2)\sqrt{\text{Re}}]. \end{aligned} \tag{59}$$

where $I_1 = 8\pi \int_0^1 (1 - x^3)x^{-1/4}/P_2(x) dx$, $\lambda_2 = s/b$, $\lambda_3 = d/b$, $\zeta = \omega/\tilde{\omega}$, $\bar{\delta} = \delta(r)/(\lambda_1 b) = [\zeta/(\text{Re}(2 + 3\zeta))]^{1/2} \cdot [150(6\zeta - 1)/(\zeta + 7/3)]^{1/4}$.

Equation (59) was solved at given parameters λ_j and Re numerically by the bisection method. For example, at $R_1 = 0.005$ m, $R_2 = 0.092$ m, $R_3 = 0.104$ m, $s = 0.031$ m, $d = 0.01$ m, $\omega = 20$ rpm, and when the gas inside the casing is air, the value $\beta_{4,4} = 1/\zeta = 0.345$ was obtained from (59) so that $\tilde{\omega} = 0.345\omega$. Subscripts 4,4 by β indicate that $N_1 = N_2 = 4$, where N_1 and N_2 are the degrees of polynomials $P_1(x)$ and $P_2(x)$ entering (30). Figure 9 shows the graphs $v = v(r_0, z)$ in two different sections $r = \text{const}$: section $r \approx 0.57(R_1 + R_2)$ lies in about the middle of the interval $[R_1, R_2]$, and section $r \approx 0.93R_2$ lies near the disc butt end. Solid lines in Fig. 9 show the new analytic solution for v at $\beta_{4,4} = 0.345$; the dashed line is the analytic solution of [15], $\beta = 0.54$. The dotted line in Fig. 9 is the numerical solution of the problem under study, which was obtained by the difference/spectral method described in Section 2 at the same input data as the analytic solution. The stationary difference solution was

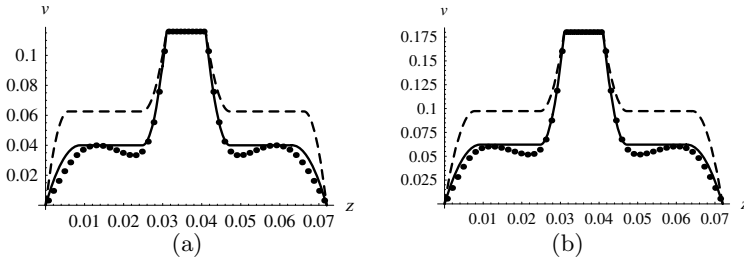


Fig. 9. Graphs of $v = v(r_0, z)$: (a) $r_0 = 0.05534 \approx 0.57(R_1 + R_2)$; (b) $r_0 = 0.08606 \approx 0.93R_2$

obtained by the steadying method on a non-uniform spatial computational grid which had 65, 41, and 51 nodes along the θ -, r -, and z -axes, respectively. The analytic solution of [15] ($\beta = 0.54$) overestimates the values of the velocity component $v(r, z)$ in the region of the quasi-solid fluid rotation.

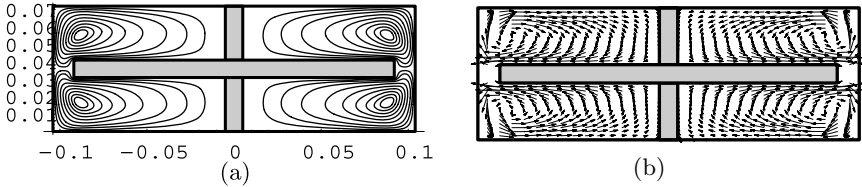


Fig. 10. Flow patterns in the casing axial section: (a) the streamlines; (b) the velocity vectors

The flow patterns presented in Fig. 10 were obtained with the aid of the difference/spectral method described in Section 2. The behaviour of streamlines agrees qualitatively with the one presented in [10].

6 Further Solutions for the Boundary Layer of the Casing Base

For the purpose of a further reduction of the value of the objective function (48) of the least-squares method we present in the following the analytic solutions for the boundary layer of the motionless casing base for the cases when $N_1 \geq 4$, $N_2 \geq 4$, $N_1 + N_2 > 8$, where N_1 and N_2 are the degrees of polynomials (31). For example, in order to obtain with the aid of our *Mathematica*-program the solution for the case $N_1 = 4$, $N_2 = 5$ it is sufficient to replace the line

$$P2 = d0 + d1*x + d2*x^2 + d3*x^3 + d4*x^4;$$

with the line

$$P2 = d0 + d1*x + d2*x^2 + d3*x^3 + d4*x^4 + d5*x^5;$$

As in the case of $N_1 = N_2 = 4$, the *Mathematica* function `FindMinimum` was used for computing the vector \mathbf{X}^* providing the minimum of the objective function.

Table 2. The values of $F(\mathbf{X}^*)$ and β_{N_1, N_2} for different N_1, N_2

N_1	4	4	5	5	5	6	6
N_2	4	5	4	5	6	5	6
$10^{-3}F(\mathbf{X}^*)$	4.51	2.56	2.08	2.05	1.48	1.33	1.32
β_{N_1, N_2}	0.345027	0.345057	0.345056	0.345045	0.345062	0.345046	0.345045

Table 2 presents the values of $F(\mathbf{X}^*)$ for different N_1 and N_2 . It can be seen that a significant reduction of the value of $F(\mathbf{X}^*)$ has occurred at $N_1 = 5, N_2 = 6$ in comparison with the case $N_1 = N_2 = 5$. However, at a further increase in N_1 and N_2 , the reduction of the optimal value of the objective function $F(\mathbf{X})$ at the point of minimum \mathbf{X}^* slows down.

We present below, by analogy with the case $N_1 = N_2 = 4$, the forms of the functions $\Phi(x)$ and $\delta_{0, N_1, N_2}(x)$ describing the gas flow in the boundary layer of the casing motionless base.

1°. $N_1 = 4, N_2 = 5$

$$\begin{aligned} \Phi_{4,5} &= x^{3/4} b \sqrt{\nu \tilde{\omega}} (1 - \lambda_1 - x) (2.856340 - 1.570379x + 1.074781x^2 + 0.082472x^3); \\ \delta_{0,4,5} &= x^{1/4} \sqrt{\nu / \tilde{\omega}} (4.185028 - 5.471901x + 3.871863x^2 - 0.997633x^3 \\ &\quad - 0.353929x^4 - 0.064053x^5). \end{aligned}$$

2°. $N_1 = 5, N_2 = 4$

$$\begin{aligned} \Phi_{5,4} &= x^{3/4} b \sqrt{\nu \tilde{\omega}} (1 - \lambda_1 - x) (2.787174 - 1.625395x + 0.964173x^2 - 0.005921x^3 \\ &\quad - 0.044119x^4); \\ \delta_{0,5,4} &= x^{1/4} \sqrt{\nu / \tilde{\omega}} (4.110059 - 5.462528x + 3.745126x^2 - 1.038496x^3 - 0.328643x^4). \end{aligned}$$

3°. $N_1 = 5, N_2 = 5$

$$\begin{aligned} \Phi_{5,5} &= x^{3/4} b \sqrt{\nu \tilde{\omega}} (1 - \lambda_1 - x) (2.810254 - 1.611966x + 0.995323x^2 + 0.014495x^3 \\ &\quad - 0.035890x^4); \\ \delta_{0,5,5} &= x^{1/4} \sqrt{\nu / \tilde{\omega}} (4.132542 - 5.483389x + 3.764532x^2 - 1.055980x^3 \\ &\quad - 0.351856x^4 - 0.015801x^5). \end{aligned}$$

4°. $N_1 = 5, N_2 = 6$

$$\begin{aligned} \Phi_{5,6} &= x^{3/4} b \sqrt{\nu \tilde{\omega}} (1 - \lambda_1 - x) (2.876983 - 1.619852x + 1.076446x^2 + 0.077721x^3 \\ &\quad - 0.007302x^4); \\ \delta_{0,5,6} &= x^{1/4} \sqrt{\nu / \tilde{\omega}} (4.188974 - 5.648875x + 3.795984x^2 - 1.197402x^3 \\ &\quad - 0.549613x^4 - 0.162125x^5 - 0.030455x^6). \end{aligned}$$

5°. $N_1 = 6, N_2 = 5$

$$\begin{aligned} \Phi_{6,5} &= x^{3/4} b \sqrt{\nu \tilde{\omega}} (1 - \lambda_1 - x) (2.844606 - 1.647739x + 1.016851x^2 + 0.021477x^3 \\ &\quad - 0.050451x^4 - 0.019380x^5); \\ \delta_{0,6,5} &= x^{1/4} \sqrt{\nu / \tilde{\omega}} (4.156011 - 5.637994x + 3.744278x^2 - 1.215339x^3 \\ &\quad - 0.552950x^4 - 0.138877x^5). \end{aligned}$$

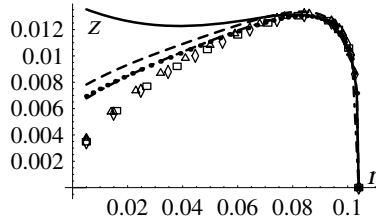


Fig. 11. The curves $z = \delta_0(r)$ by different formulas: (—) $N_1 = N_2 = 4$; (---) $N_1 = 4, N_2 = 5$; (· · ·) $N_1 = 5, N_2 = 4$; (- · - · -) $N_1 = N_2 = 5$; ($\Delta \Delta \Delta$) $N_1 = 5, N_2 = 6$; ($\square \square \square$) $N_1 = 6, N_2 = 5$, ($\diamond \diamond \diamond$) $N_1 = N_2 = 6$

6°. $N_1 = 6, N_2 = 6$

$$\begin{aligned} \Phi_{6,6} &= x^{3/4} b \sqrt{\nu \tilde{\omega}} (1 - \lambda_1 - x) (2.848314 - 1.646002x + 1.022749x^2 + 0.026754x^3 \\ &\quad - 0.046820x^4 - 0.018012x^5); \\ \delta_{0,6,6} &= x^{1/4} \sqrt{\nu / \tilde{\omega}} (4.159553 - 5.642342x + 3.748718x^2 - 1.216971x^3 \\ &\quad - 0.556802x^4 - 0.143530x^5 - 0.002778x^6). \end{aligned}$$

Table 2 also presents the values of the ratio $\beta_{N_1, N_2} = \tilde{\omega} / \omega$ corresponding to the given N_1 and N_2 . The β_{N_1, N_2} values were obtained as in the foregoing section by the numerical solution of equation (59). The geometric parameters $\lambda_1, \dots, \lambda_4$ were the same as for $N_1 = N_2 = 4$ in all the above computations for different N_1 and N_2 . It can be seen from Table 2 that the coefficient β_{N_1, N_2} depends weakly on N_1 and N_2 . But since the coefficients of equation (59) depend on $\lambda_1, \dots, \lambda_4$ and Re , at a variation of any of these parameters it is natural to expect also a variation of the coefficient $\beta_{N_1, N_2} = 1 / \zeta$.

Figure 11 shows the curves $z = \delta_{0, N_1, N_2}(r)$ for different N_1, N_2 . The value $\tilde{\omega}$ for each specific pair (N_1, N_2) was specified by formula $\tilde{\omega} = \beta_{N_1, N_2} \omega$, where the values of β_{N_1, N_2} were taken from Table 2. It can be seen in Fig. 11 that for all considered N_1, N_2 , the boundary layer thickness $\delta_0(r)$ on the motionless casing base reduces with reducing polar distance r .

7 Conclusions

We now draw several conclusions from the above.

1. The analytic solution for the flow parameters inside the boundary layer of the casing motionless base, which was obtained in [15], is inapplicable for verification of the numerical solutions obtained by the numerical integration of the three-dimensional Navier–Stokes equations (1), (2) because of the negative thickness of the boundary layer in the shaft neighborhood.

2. Several new analytic solutions are obtained for the flow parameters inside the boundary layer of the casing motionless base when the values of N_1, N_2 are within the limits $4 \leq N_1, N_2 \leq 6$. The common feature of all these solutions is that the boundary layer thickness $\delta_0(r)$ on the casing motionless base reduces with reducing polar distance r .

3. It is proposed to take into account the viscous friction force moment on the lateral casing wall. This has improved considerably the accuracy of the approximate analytic solution.

References

1. Baev, V.K., Bazhaikin, A.N., Frolov, A.D., Takeda, K., Hirano, Y.: Air cleaning from ammonia in agricultural-purpose rooms. *Ecology and Industry of Russia* 11, 13–16 (2005) (in Russian)
2. Draper, N.R., Smith, H.: *Applied Regression Analysis*. John Wiley & Sons, New York (1998)
3. Fadlun, E.A., Verzicco, R., Orlandi, P., Mohd-Yusof, J.: Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *J. Comput. Phys.* 161, 35–60 (2000)
4. Fomichev, V.P.: Device for Gas Cleaning. Patent RU No. 2229658 C2, Moscow (2004) (in Russian)
5. Khaidarov, S.V.: Experimental Investigation of Heat and Mass Exchange in Diametral Disc Fans. Ph.D. Thesis, ITAM SB RAS, Novosibirsk (2000) (in Russian)
6. Kim, J., Moin, P.: Application of a fractional-step method to incompressible Navier–Stokes equations. *J. Comp. Phys.* 59, 308–323 (1985)
7. Kiselev, S.P., Vorozhtsov, E.V., Fomin, V.M.: *Foundations of Fluid Mechanics with Applications: Problem Solving Using Mathematica*. Birkhäuser, Basel (1999)
8. Lai, M.C., Lin, W.-W., Wang, W.: A fast spectral/difference method without pole conditions for Poisson-type equations in cylindrical and spherical geometries. *IMA J. Numer. Anal.* 22, 537–548 (2002)
9. Loitsyanskii, L.G.: *Boundary Layer Aerodynamics*. Gos. izdatelstvo tekhniko-teoreticheskoi literatury, Leningrad, Moscow (1941) (in Russian)
10. Loitsyanskii, L.G.: *Laminar Boundary Layer*. In: GIFML, Moscow (1962) (in Russian)
11. Morozov, V.A., Kirsanova, N.N., Sysoev, A.F.: A complex of algorithms for fast Fourier transform of discrete series. In: *Numerical Analysis in FORTRAN*, issue 15, pp. 30–51. The Moscow State Univ., Moscow (1976) (in Russian)
12. Prikhodko, Yu.M.: Investigation of Flow and Heat Exchange in Diametral Disc Fans at Low Reynolds Numbers. Ph.D. Thesis, ITAM SB RAS, Novosibirsk (2008) (in Russian)
13. Samarskii, A.A., Gulin, A.V.: *Numerical Methods*. Nauka, Moscow (1989) (in Russian)
14. 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. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) *Computer Algebra in Scientific Computing/CASC 2004*, pp. 409–422. Techn. Univ. Munich, Munich (2004)
15. Schultz-Grunow, F.: Der Reibungswiderstand rotierender Scheiben in Gehäusen. *Zeitschr. für angew. Math. und Mech.* 15, 191–204 (1935)
16. Solomakhova, T.S. (ed.): *Centrifugal Ventilators*. Mashinostroenie, Moscow (1975) (in Russian)
17. Tesla, N.: Turbine. US Patent No. 1061206, May 6 (1913)
18. Verzicco, R., Orlandi, P.: A finite-difference scheme for three-dimensional incompressible flows in cylindrical coordinates. *J. Comput. Phys.* 123, 402–414 (1996)

Hybrid Solution of Two-Point Linear Boundary Value Problems

M. Youssef and G. Baumann

German University in Cairo,
Mathematics Department,
New Cairo City,
Egypt

Maha.Youssef@GUC.edu.eg, Gerd.Baumann@GUC.edu.eg

Abstract. We discuss a general approach to solve linear two points boundary value problems (BV) for ordinary differential equations of second and higher order. The combination of symbolic and numeric methods in a hybrid calculation allows us to derive solutions for boundary value problems in a symbolic and numeric representation. The combination of symbolic and numeric calculations simplifies not only the set up of iteration formulas which allow us to numerically represent the solution but also offers a way to standardize calculations and deliver a symbolic approximation of the solution. We use the properties of distributions and their approximations to set up interpolation formulas which are efficient and precise in the representation of solutions. In our examples we compare the exact results for our test examples with the numerical approximations to demonstrate that the solutions have an absolute error of about 10^{-12} . This order of accuracy is rarely reached by traditional numerical approaches, like sweep and shooting methods, but is within the limit of accuracy if we combine numerical methods with symbolic ones.

1 Introduction

The solution of boundary value problems is central in engineering applications such as mechanical structures, electrical circuits, hydrodynamic and thermal systems [1]. A great deal of problems is formulated as boundary value problem. There are a large number of methods currently available to solve these type of problems numerically and there seems no need to add another one. However, if we look at the approaches used in these methods we observe that results are either gained by crude brute force methods which lack accuracy but are simple to use or highly sophisticated methods which use special procedures applicable only for a specific kind of problems [5, 4, 9, 2, 3]. To bridge this gap between numerical and analytic calculations we discuss a hybrid approach which combines the clearness of analytical representation and the efficiency of numerical calculations. The approach we use is based on analytic functions and the generalization of functions to distributions.

The central function we will use in our approximations is the scaled sinc function defined by

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}. \tag{1}$$

And the shifted Sinc represented by

$$S(x, k) = \text{sinc}\left(\frac{x - kh}{h}\right) \tag{2}$$

where k is an integer number $k \in \mathbb{Z}$ and h the step length of the shift.

The idea of Sinc methods is to represent a function by Sinc approximations. A function can be approximated by using shifted Sinc functions by a linear superposition of these functions. The basic relation is given by the following expansion

$$u(x) = \sum_{k=-M_x}^{N_x} u_k \text{sinc}\left(\frac{x - kh}{h}\right) \tag{3}$$

where M_x and N_x are the lower and upper limits of the discrete approximation interval, h is the step length of the discretization and u_k represents the expansion coefficients of the function u [11]. If we deal with differential equations not only the unknown function u is needed but also derivatives of this function of different orders are needed. This means we have to handle terms including derivatives of u with respect to the independent variable x such as $\frac{d^\mu u(x)}{dx^\mu}$.

Using (3) we can represent higher order derivatives by

$$\frac{d^\mu u(x)}{dx^\mu} = \sum_{k=-M_x}^{N_x} u_k \frac{d^\mu}{dx^\mu} \left(\text{sinc}\left(\frac{x - kh}{h}\right) \right). \tag{4}$$

Equation (4) represents the higher order derivative symbolically. In fact, we are interested in a numerical representation which means we have to collocate the real line. This collocation is shown in the following relation representing a discrete version of (4).

$$\frac{d^\mu u(x)}{dx^\mu} \Big|_{x=mh} = \sum_{k=-M_x}^{N_x} u_k \frac{d^\mu}{dx^\mu} \left(\text{sinc}\left(\frac{x - kh}{h}\right) \right) \Big|_{x=mh} \tag{5}$$

with $\mu \in \mathbb{N}_0$ and $-M_x \leq m \leq N_x$.

Combining (3) and (5) in differential equations will result into a sinc representation of an ordinary differential equation. Using this truncated Sinc expansion guarantee that the error will exponentially decaying (as we will see in section 3) which is not available in other truncated expansion like truncated Fourier series expansion.

2 Methods of Calculations

To use Sinc functions to approximate derivatives in any kind of differential equation for an arbitrary order we start with the one dimensional representation of derivatives using (3) and (5) as basis to start with. First the scaled and shifted Sinc functions are defined in *Mathematica* by,

$$\text{sincScaled}[x_]:= \text{Sinc}[\pi x]$$

and the related shifted Sinc function by

$$S[x_ , k_]:= \text{sincScaled} \left[\frac{x - kh}{h} \right]$$

which are used in the definition of the derivative (5).

```

derivativeMatrixElements[orderx_List, min_List, kin_List]:=
Block[{res1, t2, h, phi, x, psi, k, m}, Fold[Plus, 0,
MapThread [(res1 =
(h#1 (∂{phi[x], #1} (sincScaled [phi[x]-kh/h]))) / .{x -> psi[mh]} / .phi[psi[x-] -> x;
t2 = Limit[res1, k -> m];
Fold[Times ,1,
Map[Apply[KroneckerDelta, #]&,
Complement[Transpose[{min, kin}], {{#2, #3}}]]]
Piecewise[{{t2/.{k -> #3, m -> #2}, #3 == #2}, {res1/.{k -> #3, m ->
#2}, #3 != #2}],
0)]&, {orderx, min, kin}]]]
    
```

The relation for derivatives delivers a symbolic expression for a piecewise function depending on the discrete indices m and k . An example for a second order derivative is given next

$$\begin{aligned} & \text{derivativeMatrixElements}(2, m, k) \\ & \left\{ \begin{array}{ll} -\frac{\pi^2}{3} & k = m \\ \pi h \left(\frac{2h^2 \sin(\frac{\pi(hm-hk)}{h})}{\pi^2(hm-hk)^3} - \frac{\sin(\frac{\pi(hm-hk)}{h})}{hm-hk} - \frac{2h \cos(\frac{\pi(hm-hk)}{h})}{\pi(hm-hk)^2} \right) & k \neq m \end{array} \right. \end{aligned}$$

The matrix representation of this function for a 3×3 discretization is,

$$\text{MatrixForm}[\text{Table}[\text{derivativeMatrixElements}(\{2\}, \{m\}, \{k\}), \{m, 1, 3\}, \{k, 1, 3\}]]$$

$$\begin{pmatrix} -\frac{\pi^2}{3} & 2 & -\frac{1}{2} \\ 2 & -\frac{\pi^2}{3} & 2 \\ -\frac{1}{2} & 2 & -\frac{\pi^2}{3} \end{pmatrix}$$

A graphical representation of the second order derivative a so called Toeplitz matrix, is shown in Figure 1. Figure 1 shows a 15×15 matrix representing a symmetric array of numerical values shown in different colors. One property of Toeplitz matrices is that they are "symmetric" about the main diagonal which is obvious from Figure 1.

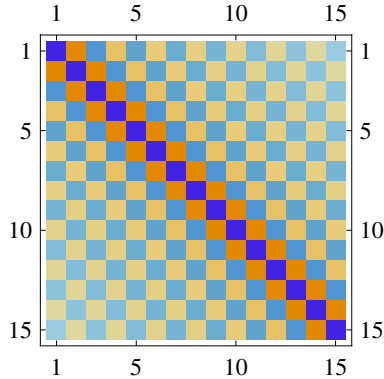


Fig. 1. Graphical representation of the content of a 15×15 Toeplitz matrix. The graphical representation shows the characteristic symmetric property.

So far we discussed only some of the important basic components of the Sinc method. In the following we will summarize the overall frame of the calculation steps. We will present the structure of the related calculation algorithm in an object oriented program based on *Elements* [15], step by step.

2.1 Representation of the Algorithm

In this section we will set up a scheme for the total calculations which can be divided into several classes of tasks. The different classes found in analyzing the algorithm are implemented in an object oriented programming environment called *Elements* developed in *Mathematica* [15]. The different classes used in the implementation are listed below. For each class we give a short description expressing which part of the calculation is represented in this part of the calculation.

1. ***BasicParameters Class***: A collection of all the influencing parameters in the calculation like α , γ , Mx , d , ax , bx , etc...
2. ***Manifold Class***: A collection of independent and dependent variables.
3. ***subvarietyOfExtendedJetBundle Class***: The subvariety of the jet space defined by the differential equation.
4. ***conformalMapping Class***: Different conformal mappings and their inverses for different types of boundary values.
5. ***sincFunctions Class***: The function used to represent the expansion basis and the representation of derivatives.

6. **collocationTransform Class:** A collection of functions that generate the discrete representation of the equation.
7. **collocationMatrices Class:** A collection of functions that create the needed matrices.
8. **EigenSolution Class:** The representation of the solution.
9. **SolutionFunction Class:** The solution of the BV problem.

The use and application of these classes is straight forward. The following example demonstrates the application of the classes and the derived objects for a specific problem. The example equation we use to demonstrate the steps of the calculation is a second order singular BV problem given by

$$\partial_{x,x}u(x) + \frac{1}{1-x^2}\partial_x u(x) + 7xu(x) = 1 - x^2 \tag{6}$$

with $u(1) = 0$ and $u(6) = 0$.

This second order equation is defined on a two dimensional manifold with variables $\{x, u\}$. This manifold is defined as an object derived from the class `Manifold`

```
man1 = manifold ◦ new[{X → {x}, U → {u}}]
      < Object of Manifold >
```

In the prolonged space a subvariety defined by the differential equation is defined by a jet bundle. In practical terms we define this jet bundle by the following derivation of an object from the class `subvarietyOfExtendedJetBundel`. This object uses the manifold object `man1` to represent the geometric structure of the space where the differential equation lives in

```
subVar1 = subVarietyOfExtendedJetBundel
         ◦ new [ { manifoldM → man1,
subvariety → { ∂x,xu[x] +  $\frac{1}{1-x^2}$ ∂xu[x] + 7xu[x] == 1 - x2 },
modelParameters → {} } ]
      < Object of SubVarietyOfExtendedJetBundel >
```

For a boundary problem we need to specify the constraints under which we are going to solve the equation. This part of information is related to the basic parameters used in the calculation. The lower bound of elements in the collocation step is set to $Mx = 8$. The boundaries are located at $x = 1$ and $x = 6$. Parameters as $\gamma = 0$ and $d = 1$ are used to guarantee an efficient convergence (see the discussion in Section 3).

```
b1 = basicParameters ◦ new[{ax → {1}, bx → {6}, Mx → {8}, d → 1, γ → 0}]
    < Object of BasicParameters >
```

The step length $h = \sqrt{\pi d / (\alpha |M_x|)}$ used in the calculation can be derived from the object of basic parameters to be

b1 ◦ **steplengthHx**[]

$$\left\{ \frac{\sqrt{\frac{\pi}{2}}}{2} \right\}$$

using default convergence parameters $\alpha = 1$ and $d = 1$. Note that the current value of h is not a small number. The interval of the boundary problem is finite and thus needs the specific conformal mapping for finite intervals to represent Dirichlet boundaries. The object of conformal mappings uses the object with basic parameters to ensure that the step length and the boundary values are used in all transformations

c1 = **conformalMapping** ◦ **new**[[**intervalObject** → **b1**, **manifoldObject** → **man1**]]

<Object of ConformalMapping>

The mapping is selected automatically depending on the type of boundary values we specify, finite or infinite. For the current example we have the conformal map and its inverse as the two transformations

c1 ◦ **conformalMap**[]

$$\left\{ \log \left[\frac{-1 + x}{6 - x} \right] \right\}$$

The inverse mapping is given by

c1 ◦ **inverseConformalMap**[]

$$\left\{ \frac{1 + 6e^x}{1 + e^x} \right\}$$

Both transformations depend on the boundary values at $a_x = 1$ and $b_x = 6$. The next step is to derive an object which contains information on *sinc* functions. The `sincFunction` class uses information contained in two objects, basic parameters and conformal mappings. An instance of the *sinc* class is thus generated by

sinc01 = **sincFunctions** ◦ **new**[[**intervalObject** → **b1**, **conformalObject** → **c1**]]

< Object of SincFunctions >

From this object we can derive for example the elements of the Toeplitz matrix for second order derivatives.

sinc01 ◦ **derivativeMatrixElements**(**2**, **j**, **k**)

$$\begin{cases} -\frac{\pi^2}{3} & k = j \\ \pi h \left(\frac{2h^2 \sin\left(\frac{\pi(hj-hk)}{h}\right)}{\pi^2(hj-hk)^3} - \frac{\sin\left(\frac{\pi(hj-hk)}{h}\right)}{hj-hk} - \frac{2h \cos\left(\frac{\pi(hj-hk)}{h}\right)}{\pi(hj-hk)^2} \right) & k \neq j \end{cases}$$

The first element of the piecewise function represents the diagonal elements of the matrix and the second line denotes the off diagonal elements of the Toeplitz matrix. The next step is to combine both branches, the differential equation and the solution representation, and derive information for the collocation transformation. The collocation transformation incorporates the jet space properties and the basic parameters which are used in the definition of this object.

```
colTr1 = collocationTransform ◦ new[{jetBundelObject → subVar1,
                                     intervalObject → b1}]
< Object of CollocationTransform >
```

Functions defined in this class are able to use the original equation and generate a discrete representation of this equation by using the transformations

```
coltr1 = colTr1 ◦ collocationTransformation[ ]
```

```
{ {x → φx[x]}, {u → Function[{x}, g1[x]u[φx[x]]]}, {g1 → Function[{x}, 1]}}
```

The transformation introduced for the dependent variable is $u = g(x)u\left(\frac{\phi(x)}{h}\right)$ which allows us to incorporate the boundaries of a conformal mapping $\phi(x)$. If the discretization is applied by using the inverse conformal mapping ψ , $x_k = \psi(hk)$, we find the discrete version of the equation.

```
deqs = colTr1 ◦ discreteSymbolicCollocationTransform[ ]
```

$$\{7u[hxm] \psi x[hxm] + \frac{u'[hxm] \phi x'[\psi x[hxm]]}{1 - \psi x[hxm]^2} +$$

$$\phi x'[\psi x[hxm]]^2 u''[hxm] + u'[hxm] \phi x''[\psi x[hxm]] == 1 - \psi x[hxm]^2\}$$

Where hx is now the step length and mx is the collocation index along the x-axis.

Knowing the symbolic equation and its discrete version we can go on to generate the matrix representation of this equation, this is generated next based on the collocation properties and the *sinc* function properties

```
colMat1 = collocationMatrices ◦ new[{collocationTrafoObject → colTr1,
                                     sincFunctionObject → sinc01}]
< Object of CollocationMatrices >
```

The collocation matrix class allows to represent the original differential equation in matrix form (result suppressed).

```
eqs1 = colMat1 ◦ replaceDerivatives[ ];
```

The next class uses the collocation matrix object to derive an instance for the collocation equations and allows to derive the solution.

eigSol = eigenSolution ◦ new[{collocationMatricesObject → colMat1}]

< Object of EigenSolution >

sol01 = eigSol ◦ solution[];

This object generates the linear collocation equations and provides a function to solve these equations. In addition the solution is represented in a symbolic way. In addition to the symbolic expansion, the coefficients are numeric results based on the solution of the linear collocation equation.

The following lines represent the total calculations combined in a single function applied to problem (6). In fact the solution is a mixture of numeric and symbolic expressions, and in combination a hybrid representation of the solution. The solution for (6)

$$\text{eq01} = \partial_{x,x}u[x] + \frac{1}{1-x^2}\partial_xu[x] + 7xu[x] == 1 - x^2$$

$$7xu[x] + \frac{u'[x]}{1-x^2} + u''[x] == 1 - x^2$$

is gained by applying BVDSolve to this equation. The total number of steps presented so far are collected in this single function BVDSolve. It generates the classes and objects as discussed and offers a single interface to the user.

sol01 = BVDSolve[{eq01}, {u}, {x}, {{1}, {6}}, {20}, 1, 0];

The gained solution can be used in a graphical evaluation of the result as shown in Figure 2.

The solution derived shows a chitter at the boundaries. This behavior is due to the low number of collocation points, 20, used in the calculation. This chitter indicates that the solution derived with this resolution is not "stable" and that the approximation is far away from the actual solution. To demonstrate that a solution becomes "stable" we increase the number of collocation points for the interval division and thus decrease the step length *h*.

sol011 = BVDSolve[{eq01}, {u}, {x}, {{1}, {6}}, {60}, 1, 0];

The result of the calculation is shown in the following plot which does not show the chitter at the endpoints of the interval as before.

Since we do not know an exact solution of this problem the question rises how to measure the accuracy of a Sinc approximation. Before we deal with this problem let us generalize the method to higher order BV-problems. In addition we will verify convergence formulas derived by Kowalski et al, Lund and Bowers [16, 17].

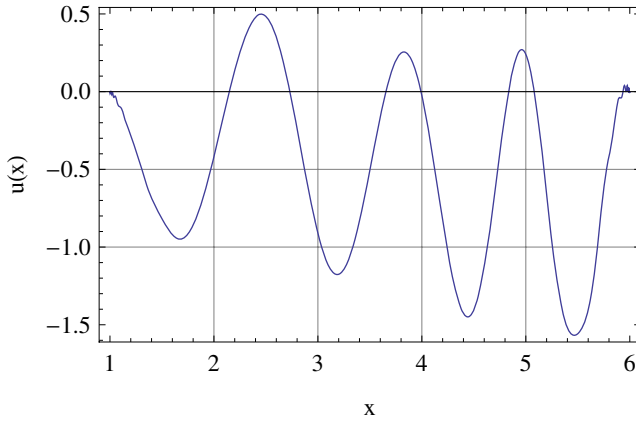


Fig. 2. Solution of the boundary value problem $\partial_{x,x}u(x) + \frac{1}{1-x^2}\partial_x u(x) + 7xu(x) = 1-x^2$ with boundary values $u(1) = u(6) = 0$

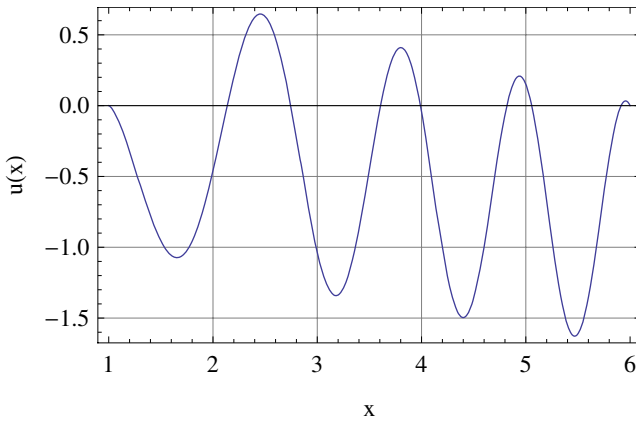


Fig. 3. Solution of the BV problem $\partial_{x,x}u(x) + \frac{1}{1-x^2}\partial_x u(x) + 7xu(x) = 1-x^2, u(1) = 0, u(6) = 0$ with a constant scaling function $g(x) = 1$ and a the lower collocation boundary with $M = 28$

2.2 Higher Order BV Problems

There is not much information available in literature for higher order boundary value problems solved by sinc-methods. For a sixth-order BV problem El-Gamel et al [13] discussed a sinc-Galerkin method. They demonstrated the application to a specific problem and discussed the convergence properties for this specific type of equation. Bialecki in his paper [9] comments on higher order BV problems and states a general formula but does not discuss details of convergence or calculations. Here we will use the already known algorithm for second order

equations and extend this algorithm to an arbitrary higher order algorithm for linear equations. The theoretical procedure is similar to the method presented for second order equations. The method for higher order is different from the second order method with respect to the incorporation of the boundaries and the discretization of derivatives. In this paper we deal only with the case where Dirichlet boundary conditions are specified, and all derivatives vanish at the boundaries. The derivatives are represented by using (5) as a general formula. The following example for a fifth order singular boundary value problem will be used to demonstrate the method.

$$\partial_{x,x}u(x) + \frac{1}{1-x^2}\partial_{x,x,x,x,x}u(x) + \frac{7x}{2-x}u(x) = 1 - x^2 \tag{7}$$

with $u(1) = 0$ and $u(6) = 0$.

We selected this equation by arbitrarily specifying the order and setting the coefficients to expressions which are singular either at the boundary at $x = 1$ or at a point in the domain of the independent variable $x = 2$. The final result of this calculation is shown in Figure 4. In *Mathematica* we define the equation by,

$$\begin{aligned} \text{eq02} &= \frac{\partial^5 u(x)}{1-x^2} + \frac{\partial^2 u(x)}{\partial x \partial x} + \frac{7xu(x)}{2-x} = 1 - x^2 \\ &\frac{u^{(5)}(x)}{1-x^2} + u''(x) + \frac{7xu(x)}{2-x} = 1 - x^2 \end{aligned}$$

and derive the solution, with the function

```
sol02 = BVDsolve[{eq02}, {u}, {x}, {{1}, {6}}, {160}, 1, 0];
```

A graphical representation of the solution with $m = 160$ follows by using the results as shown in Figure 4 (output suppressed).

To estimate an error of the discretization we use this high discretization as reference. In addition we generate three different approximations with lower discretization of the fifth order BV problem. We start with $m = 20$ and double the approximation order twice. The three solutions are compared with the approximation $m = 160$ by means of the error formula

$$\epsilon = \left| u^{(160)} - u^{(m)} \right| \tag{8}$$

For this formula we assume that $m=160$ represents the reference solution.

The three solutions follow for $m = 20$, $m = 40$, and $m = 80$ by

```
sol022 = BVDsolve[{eq02}, {u}, {x}, {{1}, {6}}, {20}, 1, 0];
```

```
sol023 = BVDsolve[{eq02}, {u}, {x}, {{1}, {6}}, {40}, 1, 0];
```

```
sol024 = BVDsolve[{eq02}, {u}, {x}, {{1}, {6}}, {80}, 1, 0];
```

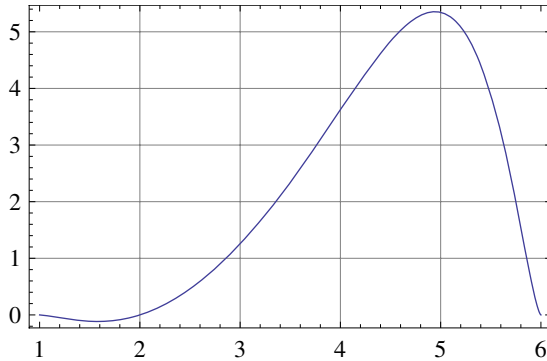



Fig. 4. Solution of the fifth order boundary value problem (30) with $m = 160$

Formula (8) is used to generate an overview of the local error. From Figure 5, it is obvious that the local error decreases if we increase the approximation order. The magnitude of the mean error at $m = 20$ is about 10^{-1} while for $m = 40$ the magnitude is 10^{-4} and with $m = 80$ we have $\epsilon \sim 10^{-6}$. This shows that the local error decreases fast if the approximation order is increased toward the reference value $m = 160$. This behavior is expected because the sinc-collocation approximation shows an exponential convergence to the true solution[11].

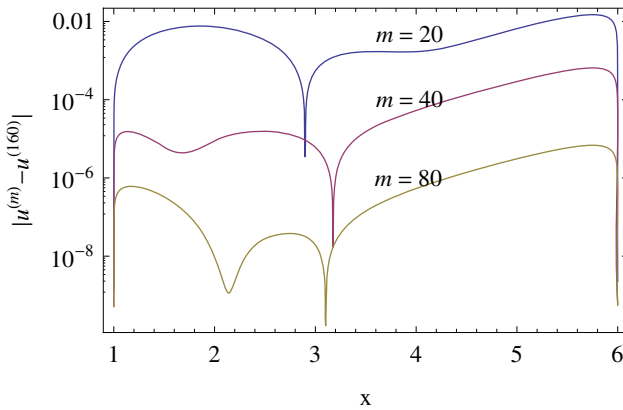


Fig. 5. Relative errors $|u^{(160)} - u^{(m)}|$ for the solution of the fifth order boundary value problem (7)

3 Convergence of the Collocation Method

To estimate the error in a Sinc calculation there is a large number of references for analytic results available [10, 17]. In practical applications it is much more

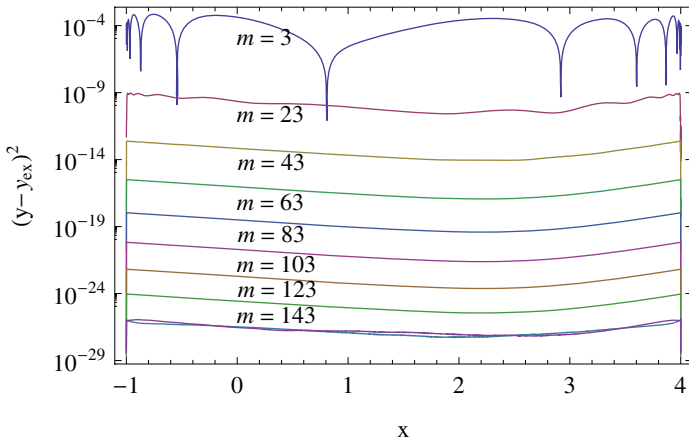


Fig. 6. Local error of the calculation based on the $(y - y_{ex})^2$ for different approximation orders $m = -M + N + 1$. The error decreases rapidly with increasing the total approximation order.

efficient to compare the convergence of a numerical procedure by changing some of the influencing parameters and measure the effect of influence. On the other hand it is much more reliable if we can compare a derived numerical solution with an exact symbolic solution. We will take the route to compare the numerical results with exact solutions knowing that an exact symbolic solution is rarely available in practical applications. As a symbolic example to demonstrate the errors generated in practical calculations we reexamine the well known example by Lybeck and Bowers [18], which is given by

$$-\partial_{x,x}y(x) + \partial_x y(x) + y(x) = \left(\frac{4}{25}\right)^2 (x^4 - 2x^3 - 29x^2 + 62x + 38) \tag{9}$$

with $y(-1) = 0$ and $y(4) = 0$.

The exact solution of this equation is given by the fourth order polynomial

$$P_4(x) = \left(\frac{4}{25}\right)^2 (x + 1)^2(x - 4)^2 \tag{10}$$

satisfying the boundary conditions. We used BVDSolve to solve the BV problem with fixed convergence parameters $\alpha = \beta = 1$ and $d = 1$. The squares of the error $(y - y_{ex})^2$ are shown in Figure 6 by changing the total number m of collocation points to see the influence of the step length h on the solution. We observe that if $m \in [3, 143]$ then the error reduces dramatically. Above $m > 160$ we again observe oscillations in the error which are due to the finite standard number representation in *Mathematica*. If the precision of the number representation in *Mathematica* is increased in the calculations the oscillations disappear for $m > 160$.

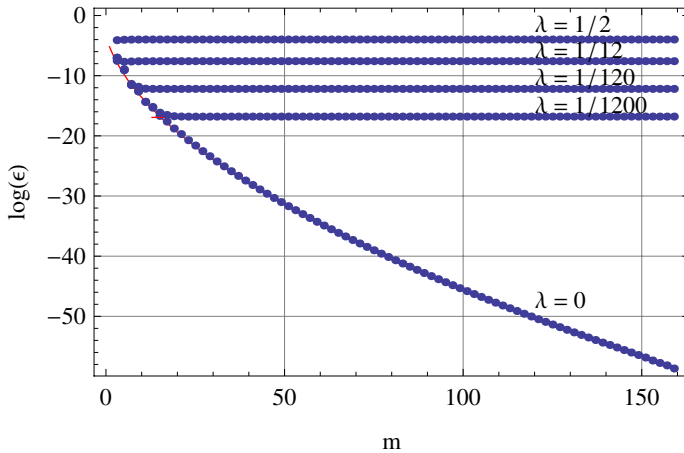


Fig. 7. Global error based on the norm $\|y - y_{\text{ex}}\|_2$ of the calculation with respect to the scaling exponent λ of the scaling function $g(x) = (\partial_x \phi(x))^{-\lambda}$

Another point of interest in estimating the error is the influence of the scaling function $g(x) = (\partial_x \phi(x))^{-\lambda}$ introduced in the class collocation. Due to the discussion in the literature [9, 11, 12], the scaling function is useful to symmetrize the numerical schemes. We examined different scaling exponents λ to see how the different scaling properties influence the error in a Sinc collocation calculation. In addition of changing the total number of collocation points we changed the scaling exponent systematically in an interval from $\lambda \in [1/2, 1/1200]$. A sub-sequence of calculations is shown in Figure 7. According to the theoretical prediction in [11] the errors should decrease exponentially like a stretched exponential function

$$\epsilon \sim \sqrt{m} e^{-\delta m^\gamma} \tag{11}$$

with a stretching exponent of $\gamma = 1/2$. If the total number of collocation points is increased the error should decrease. However, the observation in numerical calculations is that the global error defined by

$$\epsilon = \|y - y_{\text{ex}}\|^2 = \int_a^b (y - y_{\text{ex}})^2 dx \tag{12}$$

only decreases continuously if $\lambda = 0$, corresponding to a constant scaling function g . For the other cases with $0 < \lambda < 1$ we observe a plateau in a lin – log plot of the global error. This means that in principal for larger λ the global error cannot be changed by increasing the total number of collocation points but reaches a finite value which is nearly independent of m . The exponential decay of the global error is present only for $g = 1$. We verified this by fitting the gained error estimations to the function $\log(\epsilon) = -\delta m^\gamma + a + \frac{1}{2} \log(x)$ where δ , a and γ are the slope of the decay, the intercept of the error with ϵ and the deviation from the pure exponential decay. The parameters for the fit in our calculations are

$\delta = 3.93067$, $\gamma = 0.536966$, $a = -1.24904$. The deviation from the theoretical decay with $\gamma = 1/2$ is $0.036\dots$ which is quite close to the expected theoretical value.

The calculations and the fitting to the theoretical prediction show that the stretched exponential decay of the error occurs only for $\lambda = 0$ where the theoretical and experimental results are in good agreement to each other (see Fig. 7 line and dots). For all the other cases the theoretical expectation does not fit to the actual results. From a practical point of view this means that *sinc* collocation methods are most efficient for $\lambda = 0$.

4 Applications

This section contains two examples to demonstrate the behavior and the application of the method to practical engineering problems. Some of these problems are nonsingular models while the others are singular ones which are solvable by the traditional approaches only if the method is adapted to the singularity. But here we have a unique approach for both singular and nonsingular cases.

4.1 Non-uniform Bar with Distributed Force

Let us consider a standard application in mechanics to determine the elongation of a bar subject to acting forces and fixed endpoints. Such kind of problem is standard if the mass distribution and the force is constant. However, if we allow a spatial variation of the mass density and the force along the bar we have to deal with problems which may have singularities along the bar or at the endpoints of the bar. Let us consider the specific equilibrium equation

$$-\frac{d}{dx} \left(c(x) \frac{du}{dx} \right) = f(x) \tag{13}$$

for a non-uniform bar subject to homogeneous Dirichlet boundary conditions. The bar is subject to an external force f depending on the location x of action. The function $c = c(x)$ and $f = f(x)$ represents the stiffness and force distribution along the bar, respectively. For further use of this equation we define the following symbolic equation to change the two properties and derive the solution for these different models.

$$\begin{aligned} \text{eqsBar} &= -\frac{\partial}{\partial x} \left(c(x) \frac{\partial u(x)}{\partial x} \right) = f(x) \\ &-c'(x)u'(x) - c(x)u''(x) = f(x) \end{aligned}$$

where u is the deflection at distance x from the left end of the bar, and $c(x)$ and $f(x)$ are functions that depend on the applied load, the geometry of the bar, and its elastic properties. The simplest model we can derive is a bar with constant mass distribution normalized to 1 and a constant force $f = 1$ homogeneously distributed along the bar. The related equation can be derived by

$$\text{eqBar1} = \text{eqsBar}/.\{c \rightarrow (x \mapsto 1), f \rightarrow (x \mapsto 1)\}$$

$$-u''(x) = 1$$

This simple equation is solved under homogeneous Dirichlet conditions by applying the established function from above to the equation *eqBar1*. The solution in a symbolic representation follows with

$$\text{solBar1} = \text{BVDSolve}[\{\text{eqBar1}\}, \{u\}, \{x\}, \{\{0\}, \{1\}\}, \{22\}, 1, 0];$$

The solution is represented by the sinc-function expansion with numerical coefficients derived from the collocation representation of the continuous problem. So far the equation bears no problems neither in the differential equations nor at the boundaries, there are no singularities at all. The next choice for the mass distribution with $c = \log(x)/(1 - x^2)$ and $f(x) = 1$ introduces singularities at the boundaries

$$\text{eqBar2} = \text{eqsBar}/.\left\{c \rightarrow \left(x \mapsto \frac{\log(x)}{1 - x^2}\right), f \rightarrow (x \mapsto 1)\right\}$$

$$-\frac{\log(x)u''(x)}{1 - x^2} - \left(\frac{1}{x(1 - x^2)} + \frac{2x \log(x)}{(1 - x^2)^2}\right)u'(x) = 1$$

This choice generates a special kind of second order differential equation which we solve by adding the boundary conditions

$$\text{solBar2} = \text{BVDSolve}[\{\text{eqBar2}\}, \{u\}, \{x\}, \{\{0\}, \{1\}\}, \{22\}, 1, 0];$$

The two different solutions of the BV problems are shown in Figure 8. The figure shows that the boundary conditions are satisfied for the two models. Model 1 with constant densities shows a symmetric solution around the midpoint of the boundary interval as expected. Model 2 shows a negative elongation which is more pronounced in the upper part of the interval. All solutions derived with 22 collocation intervals show a continuous and smooth behavior.

4.2 Non-uniform Beam

Unlike a bar, which can only stretch longitudinally, a beam is allowed to bend. Let $0 \leq x \leq l$ represent the reference position along a horizontal beam of length l . To further simplify the model, we shall ignore stretching, and assume that the atoms in the beam can only move in the transverse direction, with $y = u(x)$ representing the vertical displacement of the atom that starts out at position x .

The strain in a beam depends on how much it is bent. Mathematically, bending is equal to the curvature of the graph of the displacement function $u(x)$, and is computed by the classical formula

$$\kappa = \frac{u''}{(1 + (u')^2)^{3/2}} \tag{14}$$

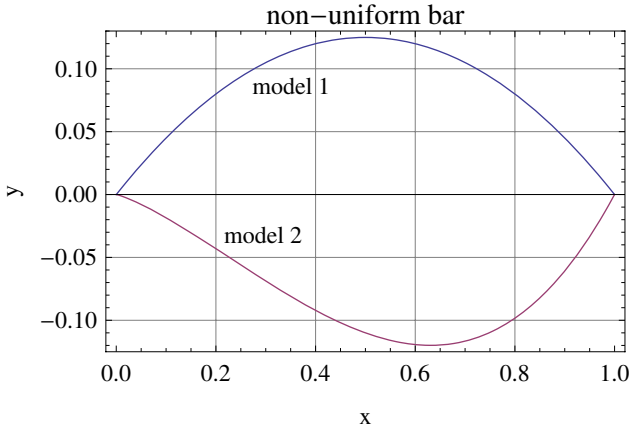


Fig. 8. Solution of the BV problem for a non-uniform bar with boundaries $u(0) = u(1) = 0$. The models have the following mass densities and forces $c = \{1, \log(x)/(1 - x^2)\}$, $f = \{1, 1\}$, respectively.

Since we are only willing to deal with linear systems, we assume that the derivative (u') $\ll 1$ of the tangent line is nearly horizontal. Then

$$\kappa \approx u'' \tag{15}$$

The next step is to formulate a constitutive relation between stress and strain. Our small bending assumption implies an elastic Hook's law relation

$$w(x) = c(x)u'' \tag{16}$$

where the proportionality factor $c(x) > 0$ measures the stiffness of the beam at point x . In particular, a uniform beam has constant stiffness, $c(x) = c$. Finally, the differential equation governing the equilibrium configuration of the beam will follow from a balance of the internal and external forces resulting to

$$\text{eqsBeam} = \frac{\partial^2 \left(c(x) \frac{\partial^2 u(x)}{\partial x \partial x} \right)}{\partial x \partial x} = f(x)$$

$$c''(x)u''(x) + 2c'(x)u^{(3)}(x) + c(x)u^{(4)}(x) = f(x)$$

We conclude that the equilibrium configuration of the beam is characterized as a solution to the fourth order ordinary differential equation.

Let us concentrate first our efforts on the uniform beam, of unit length, so $c(x) = 1$ and $f(x) = 1$. The beam equation simplifies to

$$\text{eqBeam1} = \text{eqsBeam} / \{c \rightarrow (x \mapsto 1), f \rightarrow (x \mapsto 1)\}$$

$$(u^{(4)}(x) = 1$$

Imposing the boundary conditions $u(0) = u(1) = 0$ we have

$$\text{solBeam1} = \text{BVDSolve}[\{\text{eqBeam1}\}, \{u\}, \{x\}, \{\{0\}, \{1\}\}, \{\{22\}, 1, 0\};$$

Assuming again that $f(x) = 1$ and $c(x) = 1/(1 - x^2)$ we end up with the singular equation

$$\text{eqBeam2} = \text{eqsBeam}/. \left\{ c \rightarrow \left(x \mapsto \frac{1}{1 - x^2} \right), f \rightarrow \left(x \mapsto 1 \right) \right\}$$

$$\frac{4xu^{(3)}(x)}{(1 - x^2)^2} + \frac{u^{(4)}(x)}{1 - x^2} + \left(\frac{8x^2}{(1 - x^2)^3} + \frac{2}{(1 - x^2)^2} \right) u''(x) = 1$$

Again we assume that the end points of the beam are not deflected so that we can apply the solver to this setup

$$\text{solBeam2} = \text{BVDSolve}[\{\text{eqBeam2}\}, \{u\}, \{x\}, \{\{0\}, \{1\}\}, \{\{22\}, 1, 0\};$$

The elongation of the beam with the different models for the stiffness and the external force are shown in the following Figure 9.

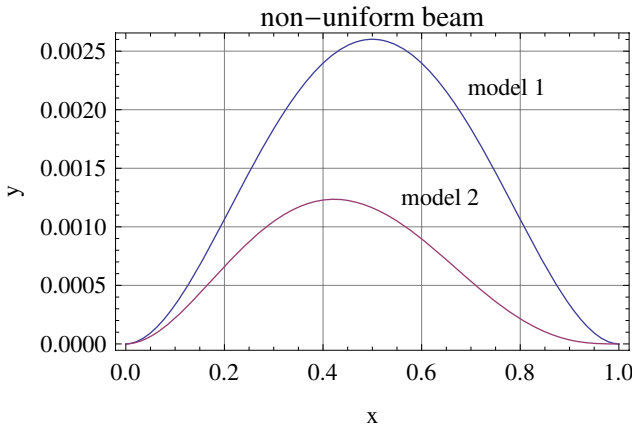


Fig. 9. Solution of the BV problem for a non-uniform beam with boundary conditions $u(0) = u(1) = 0$

As expected the homogeneous model is symmetric while the singular model shows some asymmetric shift due to the singularity.

5 Conclusions

The paper introduces a combination of symbolic and numeric calculation based on Sinc methods. The combination of symbolic and numeric calculation in a

single tool allows us to set up a procedure which represents the different steps by classes. The object oriented concept of introducing classes is naturally representing the way of calculation and allows a direct implementation of the calculation steps. The approach is not only restricted to second order BV problems but can be extended in a straight forward way to higher order equations. The solution steps based on sinc-methods guarantee an exponential convergence of the numerical calculations. This means that the number of collocation steps can be reduced to a minimal number of steps if an accuracy of a specified magnitude is required. Sinc collocation method have the property that they converge very rapidly thus we gain an efficient way to do numerical calculations.

Note: Due to the limited space all calculations in this paper have been compressed.

Acknowledgments. The Authors are very grateful to the German University in Cairo (GUC) for offering the facilities they need to complete this paper which is part of the PhD project of MY.

References

1. Duffy, D.G.: Mixed boundary value problems. Chapman & Hall/CRC, Boca Raton (2008)
2. Morlet, A.C., Lybeck, A., Bowers, K.L.: The Schwarz alternating sinc domain decomposition method. *Appl. Num. Math.* 25, 461–483 (1997)
3. Jang, A.P., Haber, S.: Numerical Indefinite Integration of Functions with Singularities. *Math. Comp.* 70, 205–221 (2000)
4. Layton, E.G.: The Fourier-grid formalism: philosophy and application to scattering problems using R-matrix theory. *J. Phys. B: At. Mol. Opt. Phys.* 26, 2501–2522 (1993)
5. Wendland, H.: Meshless Galerkin Methods using Radial Basis Functions. *Math. Comp.* 68, 1521–1531 (1999)
6. Stenger, F.: Summary of Sinc numerical methods. *J. Comp. Appl. Math.* 121, 379–420 (2000)
7. Stens, R.L.: Error estimates for sampling sums based on convolution integrals, *Inform. and Control* 45, 37–47 (1980)
8. Töplitz, O.: Zur Theorie der quadratischen und bilinearen Formen von unendlich vielen Veränderlichen. *Math. Anal.* 70, 351–376 (1911)
9. Bialecki, B.: Sinc-Collocation Methods for Two-Point Boundary Value Problems. *IMA J. Num. Anal.* 11, 357–375 (1991)
10. Stenger, F.: Matrices of Sinc methods. *J. Comp. Appl. Math.* 86, 297–310 (1997)
11. Stenger, F.: Numerical Methods Based on Sinc and Analytic Functions. Springer, New York (1993)
12. Jarratt, M.: Galerkin Schemes and the Sine-Galerkin Method for Singular Sturm-Liouville Problems. *J. Comp. Phys.* 89, 41–62 (1990)
13. El-Gamel, M., Cannon, J.R., Zayed, A.I.: Sinc-Galerkin Method for Solving Linear Sixth-Order Boundary-Value Problems. *Math. Comp.* 73, 1325–1343 (2003)

14. Narasimhan, S., Chen, K., Stenger, F.: The Harmonic-Sinc Solution of the Laplace Equation for Problems with Singularities and Semi-Infinite Domains. *Num. Heat Transf.* 33, 433–450 (1998)
15. Baumann, G., Mnuk, M.: *Elements. Math. J.* 10, 161–186 (2006)
16. Kowalski, M.A., Sikorski, K.A., Stenger, F.: *Selected topics in approximation and computation.* Oxford Univ. Press, New York (1995)
17. Lund, J., Bowers, L.K.: *Sinc methods for quadrature and differential equations,* Soc. for Industrial and Applied Mathematics, Philadelphia (1992)
18. Lybeck, N.J., Bowers, K.L.: Sinc methods for domain decomposition. *Apl. Math. Comp.* 75, 13–41 (1996)

Author Index

- Abramov, S.A. 1
- Banshchikov, Andrey V. 18
- Barkatou, M.A. 1
- Baumann, G. 373
- Berghammer, Rudolf 29
- Blinkov, Yuri A. 94
- Braßel, Bernd 29
- Bruno, Alexander 45
- Buchberger, Bruno 269
- Burlakova, Larisa A. 54
- Cheng, Jin-San 89
- Chuluunbaatar, O. 334
- Ding, Ling 66
- Edneral, Victor 45
- Faugère, Jean-Charles 79
- Flegontov, Alexander V. 81
- Gao, Xiao-Shan 89
- Gerdt, Vladimir P. 94, 106, 334
- Giesbrecht, Mark 118
- Gogin, Nikita 240
- Gusev, A.A. 334
- Inoue, Shutaro 130
- Irtegov, Valentin 142
- Karasözen, Bülent 322
- Kerber, Michael 155
- Khmelnov, D.E. 1
- Kim, Myung Sub 118
- Kitamoto, Takuya 168
- Kornyak, Vladimir V. 180
- Kragler, Robert 106
- Lasaruk, Aless 195
- Li, Jia 89
- Malaschonok, Natasha 213
- Marusina, M.J. 81
- Monagan, Michael 226
- Mylläri, Aleksandr 240
- Nagasaka, Kosaku 247
- Nemtsev, Andrew 322
- Noro, Masayuki 259
- Prokopenya, Alexander N. 106
- Regensburger, Georg 269
- Rosenkranz, Markus 269
- Rostovtsev, V.A. 334
- Saldarriaga Vargas, Clarita 284
- Schost, Éric 66
- Shemyakova, Ekaterina 299
- Sturm, Thomas 195
- Suzuki, Akira 310
- Tec, Loredana 269
- Titorenko, Tatyana 142
- Tsybulin, Vyacheslav 322
- Vinitsky, S.I. 334
- Vorozhtsov, Evgenii V. 350
- Vrbik, Paul 226
- Youssef, M. 373